Confidentiality Protection of User Data and Adaptive Resource Allocation for Managing Multiple Workflow Performance in Service-based Systems

by

Ho An

A Dissertation Presented in Partial Fulfillment
of the Requirements for the Degree
Doctor of Philosophy

Approved April 2012 by the
Graduate Supervisory Committee:

Sik-Sang Yau, Chair
Dijiang Huang
Gail-Joon Ahn
Raghu Santanam

ARIZONA STATE UNIVERSITY

May 2012

ABSTRACT

In this dissertation, two interrelated problems of service-based systems (SBS) are addressed: protecting users' data confidentiality from service providers, and managing performance of multiple workflows in SBS. Current SBSs pose serious limitations to protecting users' data confidentiality. Since users' sensitive data is sent in unencrypted forms to remote machines owned and operated by third-party service providers, there are risks of unauthorized use of the users' sensitive data by service providers. Although there are many techniques for protecting users' data from outside attackers, currently there is no effective way to protect users' sensitive data from service providers. In this dissertation, an approach is presented to protecting the confidentiality of users' data from service providers, and ensuring that service providers cannot collect users' confidential data while the data is processed or stored in cloud computing systems. The approach has four major features: (1) separation of software service providers and infrastructure service providers, (2) hiding the information of the owners of data, (3) data obfuscation, and (4) software module decomposition and distributed execution.

Since the approach to protecting users' data confidentiality includes software module decomposition and distributed execution, it is very important to effectively allocate the resource of servers in SBS to each of the software module to manage the overall performance of workflows in SBS. An approach is presented to resource allocation for SBS to adaptively allocating the system resources of servers to their software modules in runtime in order to satisfy the

performance requirements of multiple workflows in SBS. Experimental results show that the dynamic resource allocation approach can substantially increase the throughput of a SBS and the optimal resource allocation can be found in polynomial time.

ACKNOWLEDGMENTS

I would like to express my sincere gratitude to all those who helped me during my PhD study and the writing of this dissertation. Without their help and support, the completion of my PhD degree and this dissertation would not be possible. First, I would like to gratefully thank my supervisor Professor Stephen S. Yau for his help and guidance during my PhD study at ASU, especially my research and other activities in the Distributed Parallel Software Engineering (DPSE) Laboratory and the Information Assurance Center. I thank him for inspiring and encouraging me to pursue this research. His visionary thoughts and great working enthusiasm have influenced me greatly and will continuously have an impact on my future endeavors.

I also would like to thank my committee members Professors Gail-Joon Ahn, Dijiang Huang, and Raghu Santanam for their support and invaluable advice and comments on my research.

I am also indebted my colleagues in Distributed Parallel Software Engineering (DPSE) Laboratory and Information Assurance Center at ASU, especially Dazhi Huang, Yin Yin, and Jing Huang for their helps on my studies.

Last but not least, I would like to express my deep gratitude to my family for their selfless love, and all my friends for their friendliness when I am studying in a new country.

TABLE OF CONTENTS

Page

CHAPTER Page

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION AND MOTIVATION

1.1 Background on Service-Based System

Service-Oriented Architecture (SOA) is a computer system model for creating and sharing computing processes, packaged as services. Late binding, loose coupling, reusability and dynamic composition are major characteristics of SOA [1]. In SOA, a service is an independent software or hardware entity with well-defined standard interface to provide certain functions over the networks. Services are self-contained, stateless, reusable and platform-independent, which can be dynamically composed to form a workflow.

A service-based system (SBS) is a software system based on SOA. SBS is developed by invoking and coordinating various services available over networks. Although services may be distributed across networks, provided by different organizations, implemented with different techniques and executed on different platforms, the standardized interfaces and invocation protocols allow SBS developers to rapidly compose large-scale distributed software systems from existing services [2]. The interactions among services are governed by certain contracts, called the Service Level Agreements (SLA). SLA specifies certain quality of service (QoS) requirements that the service providers have to meet during the run-time of SBS. SBS have attracted much attention from industry and academia because they have the major advantage of enabling rapid development of various applications ranging from mission critical applications to simple commercial applications, including collaborative research and development, e-

business, health care, grid-enabled applications, enterprise computing infrastructures, military applications, and homeland security [3]. With the rapid growth of the Internet, SOA and virtualization technologies, SBS leads to the vision of "Internet as a supercomputer," which incorporates the concepts of "software as a service", "platform as a service", and "infrastructure as a service."

1.2 Motivation

Due to the dynamic, loosely coupled, distributed and compositional nature of SBSs, managing the quality of service (QoS) of SBS has become a major problem. A recent survey [4] shows that most of SBS users are concerned with the QoS of the SBS because the services of a SBS run on remote machines owned and operated by various third-party providers, which the users do not have any control. Hence, fundamental changes to current SBS consideration are needed to manage multiple QoS features in SBSs effectively. Among various QoS features in SBS, security and performance are two most important QoS features in SBSs [5-8]. In the following two subsections, problems of managing security and performance in current SBS are discussed.

1.2.1 Managing Security in SBS

Among the various aspects of security of SBSs, confidentiality protection of users' data in SBSs is one of most serious concerns of SBS users [8]. Although there are many studies addressing other security aspects in SBSs, such as integrity, trustworthiness, availability and reliability [9-12], currently there is no effective

approach for confidentiality protection of user's data in SBSs from service providers [4]. Current SBSs pose serious limitation on protecting users' data confidentiality. Since users' sensitive data is processed and stored at remote machines owned and operated by service providers, there are risks of unauthorized disclosure and tampering of the users' sensitive data by service providers. Although there are many techniques for confidentiality protection from outside attackers in various computing systems, such as access control, identity management, end-to-end data confidentiality and integrity assurance [13-16], these techniques cannot be applied to SBSs for confidentiality protection of users' data because they were developed only for protection from malicious parties outside the SBSs. Since service providers of SBSs are insiders of SBSs, and their threats are considered as inside threats for users' data confidentiality in SBSs, an effective approach to protecting users' data confidentiality in SBSs from service providers is needed.

1.2.2 Managing Performance of SBS

In a SBS, multiple services in different servers over networks are dynamically composed to multiple workflows. Hence, the performance of workflows in SBS is determined by the allocated system resources of each server to its services during runtime. Multiple services are often hosted by the same server in a SBS and compete for the limited system resources of the server, such as CPU-time, memory, and network bandwidth. In addition, service compositions, resource status of servers, workflow priorities and QoS requirements are usually

dynamically changing in runtime. Hence, it is necessary to have effective techniques to adaptively allocate the system resources to each service in a SBS in order to satisfy the performance requirements of multiple workflows in the SBS. The system resources considered in this dissertation is for a large-scale aggregation of dynamically scalable distributed computing resources working together over the Internet as a tremendous virtual computer [17-19]. For example, computing grids and clouds are vast resource pools for on-demand requests over the Internet. In order to allocate resources of such systems effectively, the SBS needs to have the capabilities of monitoring the changing resource status, analyzing and estimating system performance features, and adapting its resource allocation to satisfy the QoS requirements of multiple workflows simultaneously in runtime. The following challenges need to be addressed for adaptive resource allocation in runtime in a SBS:

C1) Situation Awareness: A SBS must be aware of changing situations in dynamic runtime environments. An efficient and reliable monitoring scheme is needed for collecting the relevant contextual data of situation, such as dynamic workflow compositions, the number and workload of service-requests for each workflow, performance requirements, priorities of workflows, and available system resources as well as various relevant environmental attributes.

C2) Context analysis and performance prediction: A SBS must be able to efficiently analyze the relationship between allocation of available resources and performance of workflows. An automated analysis of the monitored context data should generate good estimates of the expected QoS of workflows.

C3) Satisfying users' requirements: A SBS must be able to efficiently determine in runtime whether a resource allocation will satisfy the performance requirements of multiple workflows with dynamic situations. If some of the performance requirements of workflows cannot be satisfied, then the SBS must be able to adaptively adjust the performance of some workflows according the workflow priorities and agreements pre-defined by users.

C4) Resource adaptation: Once proper resource allocation is determined, SBS must be able to adaptively change the resource allocation to its services in runtime.

C5) Efficiency and scalability: The processes of collecting and analyzing contextual data to determine resource allocation and adapting resource allocation must be efficient and scalable

1.3 Problem Statement

- Develop an approach to protecting the confidentiality of users' data from service providers in SBSs. This approach should satisfy the following three goals: (1) Service users should be able to specify certain sensitive data not to be shared with their service providers, and make sure that the specified sensitive data is not disclosed to or tampered by their service providers even if there is no cooperation from their service providers. (2) The approach should not depend on the trust relationship among users and service providers because the trust relationship may dynamically change (3) The overhead of using this approach should not significantly degrade the overall performance of the SBS.

- Develop a resource allocation approach to adaptively allocating the system resources of servers to their services in runtime to satisfy users' performance requirements of multiple workflows in a SBS.

1.4 Presentation of this Dissertation

This dissertation is organized as follows:

The background and related research of the confidentiality protection of users' data and resource allocation in SBS will be discussed in Chapter 2. In Chapter 3, the overall approach to protecting confidentiality of users' data from service providers in SBS will be presented. One of the important features of this approach is to use data obfuscation, which will be discussed in Chapter 4. this approach will also use anonymous access to SBS by users, which will be discussed in Chapter 5. Chapter 6 will present techniques for software module decomposition and distributed execution to protect users' data confidentiality from service providers. In Chapter 7, the overall approach to adaptive resource allocation for SBSs will be presented. This approach will model the relationship between resource allocations and performance of workflows in the SBS and then use linear programming to determine resource allocation to satisfy users' requirements on performance of each workflow. Finally, a summary of the contributions presented in the dissertation and future research will be presented in Chapter 8.

CHAPTER 2

CURRENT STATE OF ART

In this chapter, the background information and existing research efforts related to the confidentiality protection of users' data in SBS and resource allocation in SBS are discussed.

2.1 Confidentiality Protection of User's Data in SBS

There are three types of problems in protection confidentiality of user's data in SBS. The first problem is the disclosure of sensitive private information when exchanging data through the services. And the sensitive private information includes: personally identifiable information, usage data, unique device identities and so on. The second problem is that people getting inappropriate or unauthorized access to personal data in the SBS by taking advantage of certain vulnerabilities, such as lack of access control enforcement, security holes and so on [20]. The third problem is that: because the feature of SBS is that it is a dynamic environment, in those service interactions can be created in a more dynamic way than traditional e-commerce scenarios [21]. Services can potentially be aggregated and changed dynamically by service providers can change the provisioning of services. In such scenarios, personal sensitive data may move around within an organization or across organizational boundaries, so adequate protection of this information must be maintained despite the changes. So design the method to protect the privacy in SBS computing must meet the dynamical exchange of data.

An approach to protecting users' data privacy using client-based privacy management in SBS was presented in [22]. In this approach, users can manage the privacy of their data in SBS through data obfuscation, privacy policy settings, auditing and monitoring of personal data in SBS. Those features are implemented as a client-side component rather than leaving them to be implemented entirely on the server side. The proposed approach can provide a user-centric trust model that helps users to control their sensitive information in SBS. However the proposed approach requires corresponding server-side components for effective operation, and hence requires some cooperation from the service provider in SBS. If there is no cooperation from service provider, this approach may not be effective to protect users' sensitive data. In addition, if the service provider is malicious, this approach cannot protect confidentiality of users' data in SBS from service providers because the service providers will have full access privilege to users' data in SBS.

An approach for managing identities in SBS that enables individual users to easily manage their own online identities and to effortlessly participate in online collaboration activities without repeated sign-on were introduced [23]. Using the identity service (or two or more different ones), users have control over who has their personal data and how it is used-minimizing the risk of identity theft and fraud. This identity service can help users manage their pseudo identities in order to hide real identity and the information about the owner of data in SBS.

Some storage-as-a service providers, such as JungleDisk, Amazon S3 and Mozy, encrypt data files with a key stored only on the user's machine. Storage-as-

a-service with no personalization can use data files encrypted in such a way that no one but the user can decrypt them (in particular, service providers cannot decrypt them). However, services which process or use some items of the data cannot use such encrypted files as input. Some such services may process input data that had been obfuscated using Voltage's Format-Preserving Encryption [24]. This encrypts specific data fields while retaining the format of data records, and preserving referential integrity. Similarly, TC3 Health Inc.'s HIPAA-compliant software pseudonymizes sensitive items before processing data using SBS [25]. However, it appears that services which calculate the sum of several data entries cannot use data encrypted using these methods as input. Hence these methods are not sufficient to deal with the database queries.

2.2 Resource Allocation in SBS

Development of QoS and resource management middleware has been studied in various research areas, such as real-time systems, distributed object-oriented systems, web services, grid computing and SBS computing. A workflow-based computational resource broker [26, 27] is presented for grid computing environment. The main function of the resource broker is to monitor the available resources and match workflow requirements to available resources over multiple administrative domains. The resource broker provides a uniform interface for accessing available system resources of computing grid via users' credentials. An open-source toolkit Globus has emerged as a standard middleware for resource management in grid computing environment [28]. Globus provides Web Service

Grid Resource Allocation and Management (WS GRAM) Protocol in which a set of web services are designed to support APIs for requesting and using grid system resources. Globus also provides Monitoring and Discovery Service (MDS) that supports a common interface for collecting contextual information on system resources, such as available processors, CPU load, network bandwidth, file system information, storage devices, and memory in computing grids.

A market-based autonomic resource management approach in SBS computing environment was developed [29], in which Service-Level Agreement Resource Allocator provides the interfaces between the SBS service provider and external users/brokers for 1) monitoring users' service requests and QoS requirements, 2) monitoring the availability of system resources, 3) examining service requests to determine whether to accept the requests according to resource availability and processing workload, 4) allocating system resources to VMs in SBS, 5) pricing the usage of resources and prioritizing the resource allocation, and 6) keeping track of the execution progress of the service requests and maintaining the actual usage of resources. This approach supports negotiation of QoS between users and providers to establish service-level agreements (SLA) and allocation of system resources to meet the SLAs.

Multi-layered resource management (MLRM) architecture using standard-based middleware technology [30] was developed for enterprise distributed real-time embedded (DRE) systems. This architecture supports dynamic resource management to optimize and reconfigure system resources at runtime in response to changing mission needs and resource status. With the dynamic resource

allocation, a DRE system can provide QoS for critical operations under the overloaded and resource constrained environments. In [31], a constraint-programming-based approach is presented to solve resource allocation problem in real-time systems. The problem of assigning a set of preemptive real-time tasks in a distributed system is formulated as a constraint satisfaction problem (CSP) with allocation, resource and timing constraints. First, the CSP is solved using constraint programming techniques to satisfy the allocation and resource constraints. Then, the solution is validated for timing constraints through Logic-based Benders decomposition [32].

In [33], a decentralized local greedy mechanism for dynamic resource allocation in web service applications was presented. In this mechanism, software agents are generated to buy and sell network services and resources to and from each other. In [34], a market based resource allocation for web services in a commercial environment was presented. In this approach, service providers employ a cluster of servers to host web services, and service consumers pay for service usage with QoS requirements, such as waiting time or response time. A framework was developed for evaluating the effect of particular resource allocation in terms of performance and average revenue earned per unit time. A heuristic algorithm was also developed for making resource allocation decisions in order to maximize revenue.

The QoS of networks has been investigated for providing prioritized services by efficient resource allocation techniques through labeling, scheduling and routing mechanisms. In [35], an optimal resource allocation and pricing

scheme for next generation multiclass networks was presented. In this scheme, an optimization problem is formulated based on a nonlinear pricing model, whose solution ensures satisfaction of network delay constraints and efficient resource allocation in dynamic multiservice networks. In [36], an energy-efficient radio resource allocation approach based on game theory for wireless networks was presented. The study shows that the game-theoretic approach for resource allocation is useful for energy-constrained wireless networks. In [37], a resource allocation scheme for the wireless multimedia applications was presented. In this scheme, an optimization problem is formulated with fairness constraints for maximizing system capacity and resource utilization. The solution of this problem yields the optimal allocation of sub-channel, path and power. A heuristic algorithm to solve the optimization problem was also presented to ensure that the adaptive resource allocation is performed efficiently in runtime. In [38], an approach to allocating resources for embedded multimedia systems using heterogeneous multiprocessors were presented, in which optimal resources are allocated to each application to meet its throughput requirement. Synchronous Dataflow Graphs (SDFG) are used to model multimedia applications with time and resource constraints and to find the optimal resource allocation.

QoS estimation according to the system activities and resource status has been studied in many ways. A QoS model of a router with feedback control that monitors the state of resource usage and adaptively adjusts parameters of traffic admission control to estimate QoS and resource utilization was presented in [39]. Batch Scheduled Admission Control (BSAC) method to predict service delay for

high priority jobs in Internet-type networks was presented in [40]. A regression-based model for dynamically provisioning resource demand to deliver given QoS expectation was presented in [41]. An adaptive model for the tradeoff between service performance and security in service-based environments was presented in [42]. This model can be used to adjust security configurations to provide sufficient protection and satisfy service performance requirements with limited system resources.

For adaptive resource allocation for QoS management in SBS, application level differentiated services [43, 44] were introduced to control QoS for different classes of service consumers. When the system resources are limited, fewer resources are allocated for normal consumers, and most resources are reserved for satisfying premium consumers' expected QoS. Feedback controlled web services [45, 46] were developed to adjust QoS to meet the most important performance when resources are limited and consumers' required performance cannot be fully satisfied. In [47], an integrated QoS management approach in SBS in order to satisfy user's QoS requirements by providing differentiated system resources and priority of workflows was presented. In [48, 49], a general methodology for developing SBS with QoS monitoring and adaptation was introduced. This methodology supports monitoring of the changing system status, analysis and control of tradeoffs among multiple QoS features, and adapting its service configuration to satisfy multiple QoS requirements simultaneously.

Existing resource allocation approaches cannot support dynamically changing runtime environments in SBS, such as workflow composition, QoS

requirements, workflow priorities and resource status. Hence, a new approach to dynamic resource allocation for SBS is needed to address the challenges discussed in Chapter 1.

CHAPTER 3

OVERALL APPROACH TO PROTECTING CONFIDENTIALITY OF USER DATA IN SBS

In this Chapter, the overall approach to protecting confidentiality of users' data from service providers in SBS will be presented. This approach has the following goals:

- Service users can make sure that their sensitive data, which the users specify not to be shared with their service providers, is not disclosed to their service providers even if there is no cooperation from their service providers
- The protection of users' confidential data does not depend on dynamic trust relationship among users and service providers.
- This approach does not cause much overhead on service performance.

3.1 Protection of Users' Data Confidentiality from Service Providers in SBS

In [50], *confidentiality* is defined as the assurance that sensitive information is not disclosed to unauthorized persons, processes, or devices. Hence, the users' confidential data, which the users do not want to be accessed by service providers should not be disclosed to service providers in SBS, including applications, platforms, CPU and physical memories.

It is noted that users' confidential data is disclosed to a service provider only if all of the following two conditions are satisfied simultaneously:

Condition 1) The service provider knows where the users' confidential data is located in SBS and has the privilege to access and collect the users' confidential data in SBS.

Condition 2) The service provider can understand the meaning of the users' data.

This is due to the following reasons: In order to collect users' data, the service provider must know the location of the data in SBS and have the privilege to access the data. Even if the service provider can collect users' data successfully, the service provider may not be able to understand the meaning of the data unless the service provider has at the least some of the following information to understand the meanings of the data:

- Types of data
- Functionalities and interfaces of the application using the data
- Format of the data

Hence, if a SBS can prevent the service providers from satisfying all the above two conditions, the SBS can protect the confidentiality of users' data in SBS from the service providers.

3.2 Problems with the Current SBS Architecture

The current SBS consists of three layers: software layer, platform layer and infrastructure layer, as shown in Figure 1. The software layer provides the interfaces for users to use service provider's applications. The platform layer provides the operating environment for the software to run using system resources. The infrastructure layer provides the hardware resources for computing, storage,

and networks. Platforms or infrastructures can be provided as virtual machines. The following are the major problems of current SBS:

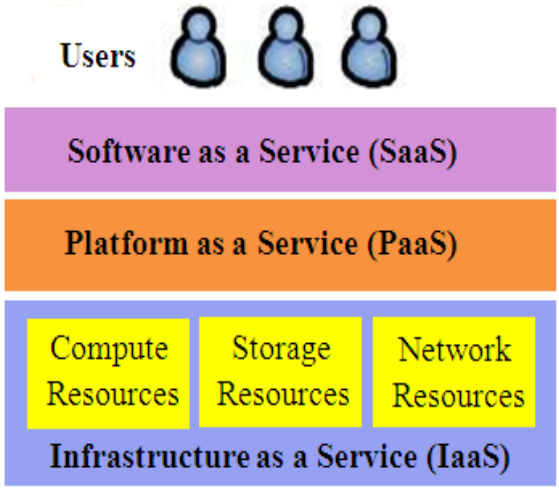

Figure 1: Current Architecture of SBS

- Each service provider has its own software layer, platform layer and infrastructure layer. When a user uses a SBS application from a service provider, the user is forced to use the platform and infrastructure provided by the same service provider, and hence the service provider knows where the users' data is located and has full access privileges to the data.
- The user is forced to use the interfaces provided by the service provider, and users' data has to be in a fixed format specified by the service provider, and hence the service provider knows all the information required for understanding users' data.

Therefore, current SBSs cannot prevent service providers from satisfying all of the two conditions.

3.3 Overall Approach for User's Data Confidentiality Protection in Service-Based Systems

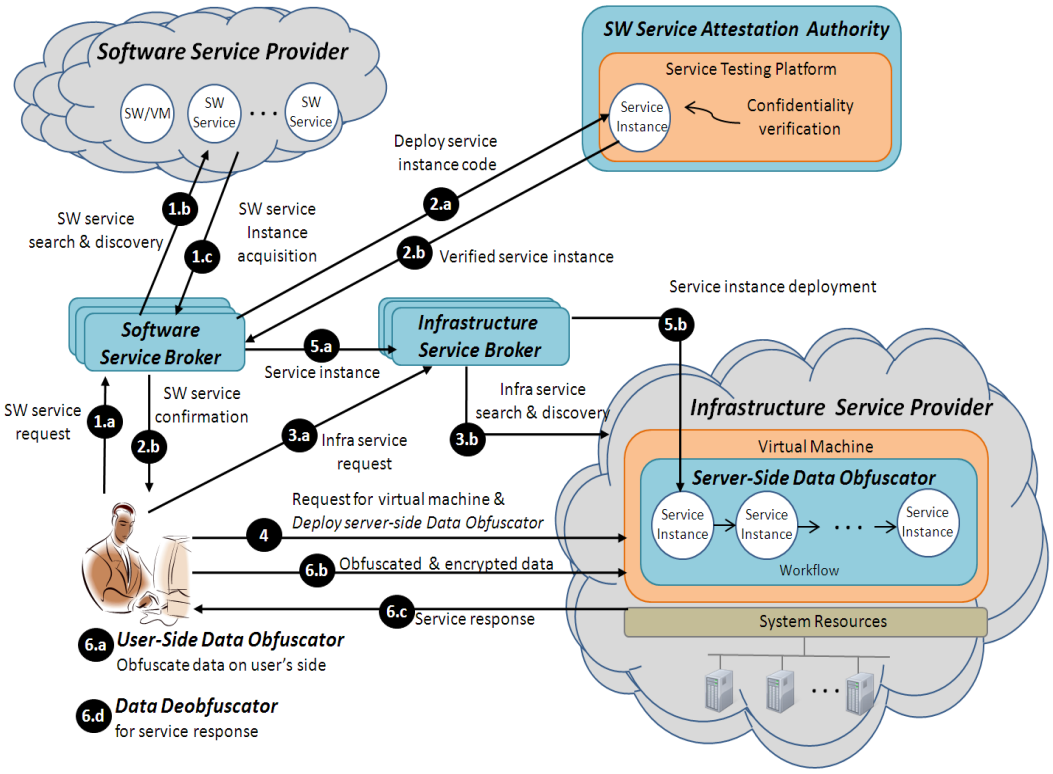

Figure 2: Overall Approach to Protecting Confidentiality of Users' Data in SBS

The overall approach for user's data confidentiality protection in SBSs can be depicted in Figure 2. This approach has the following seven entities to protect the confidentiality of users' data processed and stored in a SBS: software SBS, infrastructure SBS, software service broker, infrastructure service broker, software service attestation authority, data obfuscator and data de-obfuscator [51]. The software SBS and infrastructure SBS have the same features of the software layer in the ordinary SBS architecture. However, in this approach, the software layer and infrastructure layer are managed by different service providers. The software

service brokers and infrastructure service brokers have the same functionality of the service brokers in SOA, but they have the additional function for identity-anonymization. This approach also has the additional three entities, software service attestation authority, data obfuscator and data de-obfuscator, for protecting the users' data confidentiality. This approach will make sure that none of these entities in SBS will satisfy the two conditions simultaneously in Section 3.1. The description of each of these entities is as follows:

- **Software SBS**: A *Software SBS* provides the software as a service upon users' requests. Each software SBS may contain multiple software services, and each software service can be discovered and accessed by users through Software Service Broker. An authenticated user with proper credentials [52] can request and acquire a service instance from the software SBS. A *service instance* is a piece of compiled executable code. The executable code can be deployed and run on any Infrastructure SBS. In order to protect the intellectual property of the software, the code is compiled using various code obfuscation technologies [53, 54] so that reverse engineering on the service instance is computationally infeasible. This implies that infrastructure service providers cannot understand the functionality and algorithms of the service instance by examining at the code when the service instance is running on *Infrastructure SBS*.
- **Infrastructure SBS**: An *Infrastructure SBS* provides virtualized system resources, such as CPU, memory, and network resources. An authenticated

user can request a virtual machine on which the user can deploy any platform or operating system to execute a software service instance.

- **Software Service Broker**: A *Software Service Broker* has two major functions. First, it helps users automatically discover and access available software services. Second, it helps users hide their identities from software SBS service providers. A Software Service Broker provides identity anonymization service, by which users can use pseudonyms instead of their true identities so that the users can acquire service instances without revealing their identities. The anonymization of user identities is very important for protecting the confidentiality of users' data because the information about the data owners may reveal a lot of sensitive information regarding the data.
- **Infrastructure Service Broker**: An *Infrastructure Service Broker* has two major functions similar to a Software Service Broker. It helps users automatically discover and use available infrastructure services. It also provides identity anonymization service to prevent the system from revealing users' true identities.
- **The Software Service Attestation Authority (SSAA)**: The *SSAA* is a third party authority to verify that a service instance does not perform any malicious activity that may disclose users' confidential data. For example, a software service developer may have injected a hidden function on the software service which transmits user's confidential data to an unauthorized third party during its processing without the user's consent. Since users do not know whether a service instance will act as described in the service

description, the SSAA needs to help users test the service instance before users using it. When a service instance is deployed on the Service Testing Platform of SSAA, the SSAA tests whether the service instance performs exactly what the service provider claims, and whether the service instance may transmit users' data to any unauthorized entity.

The testing can be done by *i)* verifying whether the service instance satisfies the web service description language (WSDL) specification of the service, and *ii)* monitoring all the network traffics the software service produces during its processing. An approach to automated web service testing based on syntactic and semantic analysis of WSDL specification was presented in [55]. After completing the testing of the service instance, SSAA issues a digital certificate for the tested service instance. A certificate is attached to the service instance so that users will know whether the service instance has passed the SSAA's testing.

- **Data Obfuscator**: A *Data Obfuscator* is a middleware provided by a user that can be deployed on a virtual machine in an Infrastructure SBS. The service instance in an Infrastructure SBS can use system resources only through the interfaces of the Data Obfuscator. The Data Obfuscator consists of two modules: user-side obfuscator and server-side obfuscator. The two modules work together to provide the following security functionality to ensure that users' confidential data is not disclosed to infrastructure service providers:

i) Sets up an encryption key with the user. The key is chosen by the user and not revealed to any process, platform or device of the SBS computing system.

ii) Encrypts any data being stored in the physical storage of the SBS computing system or being transmitted through the network.

iii) Obfuscates users' sensitive data being processed in the service instances. The obfuscated data cannot be de-obfuscated in the platforms or physical devices of an Infrastructure SBS so that its infrastructure provider cannot understand the meaning of the users' sensitive data. The details of the data obfuscation will be discussed in Chapter 4.

- **Data De-obfuscator**: A *Data De-obfuscator* de-obfuscates obfuscated data so that a user can see the plain data. A Data De-obfuscator remains in the user's personal computer all the time.

This approach will have the following four features to protect confidentiality of users' data: 1) separation of software service providers and infrastructure service providers in SBS, 2) anonymous access to SBS by users, 3) data obfuscation and 4) software module decomposition and distributed execution of the modules in distinct infrastructures of service providers. With these four features, this approach can ensure that at least one of the two conditions in Section 3.1 is not satisfied in each of the seven entities in SBS computing system due to the following reasons:

- Although a software service provider knows the functionality of a service instance and the data format of users' input and can satisfy *Condition 2),* since

software service instances are deployed to Infrastructure SBS through its Software Service Broker and Infrastructure Service Broker, the software service provider does not know where users' data is located, and does not have the privilege to access the data. Thus, the software service provider cannot satisfy *Condition 1).*

- Although an infrastructure service provider knows the locations of users' specific confidential data and has the privilege to access the data being processed and stored in the infrastructure SBS, the infrastructure service provider cannot understand the meaning of the data because *i)* he/she does not know the functionality of the deployed software instance and the format of the users' specific confidential data, *ii)* the deployed software instance cannot be reverse-engineered, *iii)* the infrastructure service provider does not know the identity of the data owners, and *iv)* the users' specific confidential data is obfuscated while processed and stored in the infrastructure. Hence, the infrastructure service provider cannot satisfy *Condition 2).*
- Because the Software Service Broker does not know where users' data is located, and does not have the privilege to access user's data, the Software Service Broker cannot satisfy *Condition 1).*
- Because an Infrastructure Service Broker does not know the functionality of deployed software instance, the data format of user's data, and does not have the privilege to access the data, the Infrastructure Service Broker cannot satisfy *Condition 1).*

Overall approach is depicted in Figure 2, where the numbers are corresponding to the steps. It can be summarized as follows:

An Approach for Protecting Users' Data Confidentiality in SBS

S1) *a)* A user requests any Software Service Broker to find a software service by providing the specification of the software service [56]. *b)* The Software Service Broker performs automatic service discovery [57] to find a service instance in the Software SBS that satisfies the user's requested service requirement specification. *c)* The Software Service Broker acquires the discovered software instance using an anonymous credential [58].

S2) *a)* The Software Service Broker deploys the acquired service instance to the testing platform of a SSAA. The SSAA verifies whether the service instance performs according to its description, and the service instance does not transmit users' specific confidential data to any unauthorized entity. *b)* After the verification procedure, the software service instance is sent back to the Software Service Broker.

S3) *a)* The user asks the Infrastructure Service Broker to find an infrastructure service compatible to the service instance. *b)* The Infrastructure Service Broker discovers an infrastructure service provider, who has the capability to execute the acquired software service instance.

S4) The user requests the infrastructure service provider to set up a virtual machine and then deploys the Data Obfuscator on the virtual machine using the Agent Deployment Plans (ADPs), for automated middleware deployment and migration in service-based systems according to [59].

S5) *a)* The service instance acquired in *S1)* is sent to the Infrastructure Service Broker. *b*) The service instance is deployed on the workflow of the Data Obfuscator set up in *S4).*

S6) *a)* The user obfuscates his/her data using the user-side data obfuscator *b)* The user sends his/her obfuscated data to the workflow to process. During the processing of users' input data, the user's specific confidential data is obfuscated so that the infrastructure service provider cannot understand the meaning of user's data. *c)* After completing the processing, the processed data is sent to the user. *d)* The processed data is de-obfuscated to plain data in the user's computer.

The details of the data obfuscation will be discussed in Chapter 4. Details on the anonymization of users' identities will be discussed in Chapter 5. Details on the partitioning users' data set and controlling the data flows in SBS will be discussed in Chapter 6. Details on the resource allocation for manage performance requirements of multiple workflows will be discussed in Chapter 7.

CHAPTER 4

DATA OBFUSCATION

Data obfuscation is the process of transforming the format or structure of data to hide the meaning of data. The major difference between encryption and obfuscation is that encrypted data cannot be processed until it is decrypted, but obfuscated data can be processed without de-obfuscation. Data obfuscation is used to process users' data in an infrastructure SBS without revealing any users' specific confidential data to the infrastructure service providers.

The general algebraic description of data obfuscation is as follows. Suppose a user wishes to use an infrastructure SBS service to process a function F on the user's input x without revealing the meaning of x to the infrastructure service provider. Obfuscation function O and de-obfuscation function D have following properties:

- ***D(F(O(x, k), k)*** = ***F(x)***, where ***k*** is an obfuscation key
- The infrastructure service provider cannot understand the meaning of ***x*** by examining ***O(x, k)***
- ***D*** or ***k*** cannot be derived from ***O***
- ***O(x, k)*** and ***D(F(O(x, k), k)*** can be calculated in polynomial time

4.1 An Approach to Data Obfuscation

When a data obfuscator is deployed by a user on a virtual machine in the Infrastructure SBS, the data obfuscator obfuscates the user's input data ***x*** using one of the following three methods:

Method A) **Take dummy input data from a user and generate arbitrary dummy outputs**. In this approach, a user's input data is entered to a software service instance through the data obfuscator. Since the infrastructure service provider does not know the input data format of the service instance, the data obfuscator can take any number of dummy input data from the users. Only the user's inputs to be processed are given to the software service instance, and the data obfuscator generates arbitrary dummy outputs on the user's dummy inputs. The outputs generated by the service instance from the user's inputs, and the dummy outputs generated from the user's dummy inputs by the data obfuscator are mixed and encrypted together, and sent back to the user. The dummy outputs are marked so that data de-obfuscator in the user's computer can recognize the dummy outputs after the all the outputs of the service instance are decrypted.

For example, given user's input data $\boldsymbol{x}$, a software service function $\boldsymbol{F}$ and a secret integer key $\boldsymbol{k}$, $\boldsymbol{O}(\boldsymbol{x}, \boldsymbol{k})$ generates dummy input data $\boldsymbol{a_1, a_2 \ldots a_n}$, where $\boldsymbol{n} > \boldsymbol{k}$. For each dummy input data generated, the data obfuscator processes dummy functions $\boldsymbol{f_1(a_1)}$, $\boldsymbol{f_2(a_2)} \ldots \boldsymbol{f_n(a_n)}$ while the software service instance processes $\boldsymbol{F(x)}$. Then, an array of output data ($\boldsymbol{f_1(a_1)}$, $\boldsymbol{f_2(a_2)} \ldots$ $\boldsymbol{f_{k-1}(a_{k-1})}$, $\boldsymbol{F(x)}$, $\boldsymbol{f_k(a_k)}, \ldots, \boldsymbol{f_n(a_n)}$) is sent to the user. The data-de-obfuscator at the user's computer de-obfuscates the array of output data using the secret key $\boldsymbol{k}$. Hence, $\boldsymbol{D}((\boldsymbol{f_1(a_1)}, \boldsymbol{f_2(a_2)} \ldots \boldsymbol{f_{k-1}(a_{k-1})}, \boldsymbol{F(x)}, \boldsymbol{f_k(a_k)}, \ldots, \boldsymbol{f_n(a_n)}) = \boldsymbol{F(x)}$.

Method B) **Use a file system not known to all the infrastructure providers**. A file system is a method of storing and organizing files and data into computer

memories and storage devices, such as hard disks or CD-ROMs. If an infrastructure service provider can understand the file system structure of a user's operating system running on the infrastructure SBS, the service provider may be able to locate and extract users' data from memories or storage devices. Data obfuscator uses a file system which is not known to any of the infrastructure providers so that the infrastructure service providers cannot extract meaningful data from the memory or storage devices of the SBS computing system.

For example, data obfuscator can change the number of disk sectors in a cluster or the format of the file allocation table in the file system so that the infrastructure service provider cannot understand the file system structure. In order to prevent the service providers from extracting the users' specific confidential data from network traffics, the user's specific confidential data should be encrypted in the user's computer before being sent to service provider's infrastructure. The encrypted data is decrypted and processed on the data obfuscator's file system.

Method C) **Transform users' input data.** Data obfuscator transforms the format or value of users' input data so that the infrastructure service provider cannot understand the meaning of the data. All the operations associated with the original data must also be applicable to the transformed data so that the software service instance is able to process the obfuscated data.

For example, the data obfuscator transforms a regular string encoded with ASCII code into a different string format using a secret string encoder. Then, the

string encoded with the secret string encoder must support all the operations associated with string data, such as concatenation, split, length calculation, comparison, uppercase, lowercase or character replacement.

The general algebraic description of data transformation for obfuscation is as follows. Given input data ***x,*** a set of associated operations{$P_1,\ldots, P_n$}, an obfuscation function ***O***, a secret key ***k,*** and de-obfuscation function ***D***:

$$P_i(x) = D(P_i(O(x,k)),k), \text{ where } 1 \leq i \leq n$$

Both numerical data and text data can be obfuscated using the data transformation method.

- For numeric data, an arithmetic operation is performed on user's input data with secret integer ***k*** to transform the value of the data.

  For example, suppose a user wish to process a function $F(a,b) = a + b$, where ***a*** and ***b*** are integer values, in an Infrastructure SBS without revealing the value of ***a*** and ***b*** to the infrastructure service provider. In order to obfuscate ***a*** and ***b***, both ***a*** and ***b*** are multiplied by ***k***. Then $F(ak,bk) = (ak + bk)$ is processed in the infrastructure SBS. The infrastructure service provider cannot find the value of ***a*** and ***b*** without knowing ***k***. The user can de-obfuscate ($ak + bk$) by dividing it by ***k*** and then get the value of $a + b$. Table1 shows obfuscation and de-obfuscation functions for each arithmetic operation for numeric data.

Table 1: Obfuscation and De-obfuscation Functions for Arithmetic Operations

Operation	User input	Obfuscation function	Obfuscated data	De-obfuscation function	De-obfuscated data
Addition	a , b	$O(x,k) = xk$	ak, bk	$D(x,k) = x/k$	$a + b$
Subtraction	a , b	$O(x,k) = xk$	ak, bk	$D(x,k) = x/k$	$a - b$
Multiplication	a , b	$O(x,k) = xk$	ak, bk	$D(x,k) = x/k^2$	$a * b$
Division	a , b	$O(x,k) = x^k$	a^k, b^k	$D(x,k) = \sqrt[k]{x}$	a / b

- For text data, a secret encoder is used to transform the format of the text data. A secret encoder can be generated by randomly shuffling the order of a standard text encoding table, such as ASCII code table. Once a secret encoder is generated by a user, the data obfuscator transforms each character of the user's text data using the secret encoder. Since the transformation is done character by character, any operation associated with original text data can be applied to the transformed data. After the transformed text data is processed in the infrastructure SBS, the data is sent to the user and de-obfuscated in the user's computer. During the de-obfuscation, each character of the transformed text data is mapped to the original ASCII code.

  For example, suppose a user wants to concatenate two strings "ab" and "cd" in an infrastructure SBS. The user generates a secret encoder by randomly shuffling the ASCII code table. Table2 shows the standard ASCII code and an example of secret code generated by the user. Data obfuscator transforms "ab" and "cd" to "0111101100100111" and "1011000001111011" respectively using the secret code. The transformed strings are concatenated into "01111011001001111011000001111011" in the infrastructure SBS and

sent to the user. Data de-obfuscator in the user's computer performs the mapping of the transformed text data to the original ASCII code so that the de-obfuscated data is mapped to "01100001011000100110001101100100", which means "abcd" in the original ASCII code.

Table 2: Standard ASCII Code and a Secret Code Generated by User

Character	Standard ASCII code	A secret code generated by user
a	01100001	01111011
b	01100010	00100111
c	01100011	10110000
d	01100100	01111011

ASCII code has 127 characters in its encoding table, and each character is represented by an 8-bits binary number. By shuffling the order of the 127 characters in the table, the factorial of 127 ($\approx 3 * 10^{213}$) secret encoding tables can be generated. Thus, it is computationally infeasible for infrastructure service providers to find the meaning of the transformed text data using brute forcing attack.

Methods A) and C) are used for obfuscating and de-obfuscating users' specific confidential data during the data processing time. *Method C)* is suitable for processing numerical and text data, and *method A)* is suitable for other types of data, such as image and video data. *Method B)* is used for obfuscating and de-obfuscating any types of data during the storing time.

4.2 An Illustrative Example

This section presents an example to illustrate how my approach can protect users' specific confidential data in SBS computing systems.

Consider a group of users in different locations, who would like to have an online conference using SBS computing shown in Figure 3. They must be able to communicate with each other through voice, video, and/or instance messaging. They also need to share files with each other. To achieve this, they need to use the five services: Voice Communication Service, Video Communication Service, File Sharing Service, Instant Messaging Service, and Conference Controller. Since the voice data, video data, messages, or files may contain the specific confidential information which the users do not want the service providers to know, the users require the protection of such specific confidential data from the service providers. The virtual machine for online conferencing can be set up according to my approach as follows:

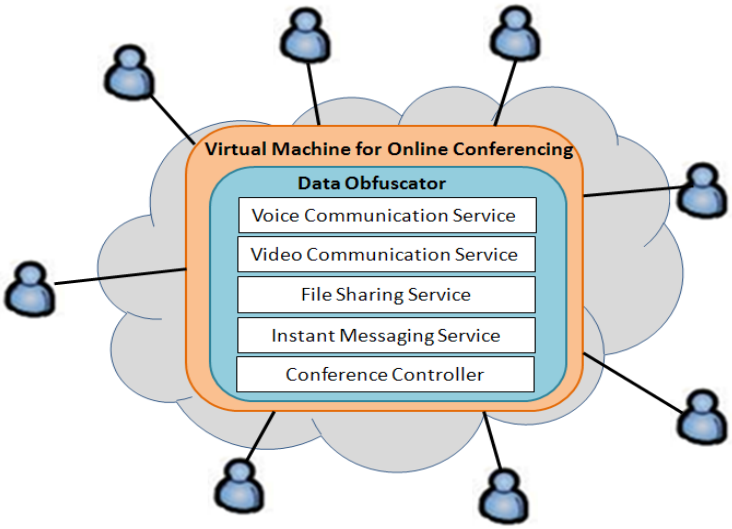

Figure 3: An Example of Online Video Conferencing

S1) *i)* The leader of the group requests a Software Service Broker to find the five software services: Voice Communication Service, Video Communication Service,

File Sharing Service, Instant Messaging Service and Conference Controller. *ii)* The Software Service Broker discovers the services. *iii)* The Software Service Broker downloads the service instances of the five software services.

S2) *i)* The Software Service Broker deploys the service instances to the testing platform of an SSAA. *ii)* The SSAA verifies the software service instances.

S3) *i)* The leader of the group requests an Infrastructure Service Broker to find an infrastructure service compatible to the service instances. *ii)* The Infrastructure Service Broker discovers an infrastructure service.

S4) A virtual machine is set up in the Infrastructure SBS. The leader of the group deploys the Data Obfuscator on the virtual machine. Obfuscation keys and encryption keys unknown to infrastructure service providers are sent to all other users.

S5) *i)* The service instances are sent to the Infrastructure Service Broker. *ii)* The service instances are deployed on the Data Obfuscator. The five service instances are composed to a workflow. The workflow provides all the functionality for online conferencing.

***S6*)** *i)* The users of the group send their input data to the workflow to process. During the processing of the users' input data, the users' specific confidential input data is obfuscated. After completing the processing, a service response of the workflow is sent to all the users of the group that the processing of their input data has been completed. *ii)* The service response of the users' specific confidential input data is de-obfuscated.

The data obfuscation during the online conferencing is done as follows:

- All users' names are transformed to pseudonyms in the Infrastructure SBS. The pseudonyms are de- obfuscated to the plain names in users' personal computers.
- All users' voice data is transformed using a secret audio encoder.
- All users' video data is transformed using a secret video encoder.
- All users' text messages are transformed using a secret string encoder.
- Dummy input data is generated frequently and sent to all the users.

In this example, at least one of the two conditions in Section 3.1 is not satisfied in each entity in the SBS computing system due to the following reasons:

- Software service providers do not know where users' data is located, and do not have the privilege to access the data. Hence, *Condition 1)* cannot be satisfied by the software service providers
- The infrastructure service provider cannot understand the meaning of users' specific confidential input data because *i)* the infrastructure service provider does not know the functionality of the deployed software instances and the data formats of the users' specific confidential input data, *ii)* the deployed software instance cannot be reverse-engineered, *iii)* the infrastructure service provider does not know the identities of the data owners, and *iv)* users' specific confidential input data is obfuscated. Hence, *Condition 2)* cannot be satisfied by the infrastructure service provider.
- The Software Service Broker does not know where users' data is located, and

does not have the privilege to access the data. Hence, *Condition 1)* cannot be satisfied by the Software Service Broker.

- The Infrastructure Service Broker does not know the functionality of the deployed software instances, the format of users' specific confidential input data, and does not have the privilege to access such data. Hence, *Conditions 1)* and 2*)* cannot be satisfied by the Infrastructure Service Broker.

Hence, the confidentiality of users' specific confidential data is protected in this example.

4.3 An Experiment on Performance of Data Obfuscator and Data De-obfuscator

I implemented a SBS service using C# Microsoft .NET framework for collaborative online documentation. Using this SBS service, multiple users can collaboratively create and edit a document within a SBS computing system. Since the document may contain confidential information, the SBS service provider must not be able to understand the meaning of the document while the document is processed in the SBS computing system.

I also implemented *Method C)* of Data Obfuscator and De-obfuscator discussed in Section 4.1. A virtual machine was set up using Microsoft Hyper-V Manager 6.0, which had 0.7 GHZ CPU, 256 MB memory and Windows Server 2003OS. The collaborative online documentation SBS service and data obfuscator are deployed on the virtual machine, and the data De-obfuscator is deployed on the user's computer as shown in Figure 4.

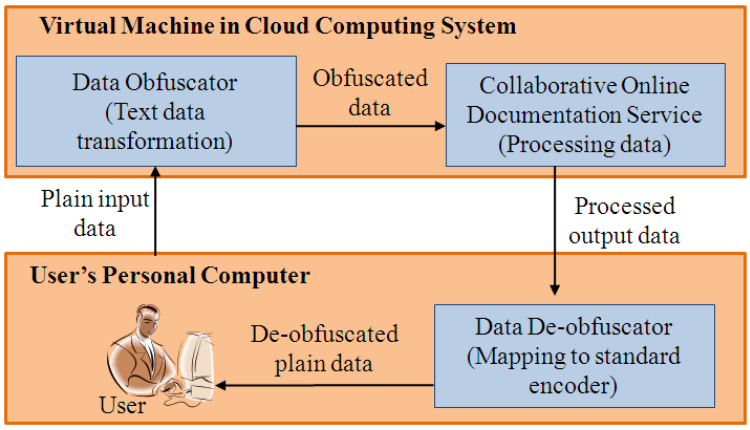

Figure 4: Collaborative Online Documentation Service with Data Obfuscator and De-obfuscator

In order to measure the performance of *Method C)* of data obfuscator and de-obfuscator, I ran two sets of experiments. In both sets of experiments, a set of input texts with various sizes are sent to the SBS service for processing and the service response time for each input was measured. In the first set of experiments, I used the SBS service without using the Data Obfuscator and De-obfuscator. In the second set of experiments, I used the SBS service along with *Method C)* of Data Obfuscator and De-obfuscator. 100% of the users' input text was specified as confidential data so that all of the input data is obfuscated and de-obfuscated during the service processing. The measured service response time for each set of experiment is shown in Figure 5.

Experimental result shows that the service response time always increases linearly as the size of the input text increases in both sets of experiments. This means that the additional overhead for data obfuscation and de-obfuscation appears to be proportional to the size of the input text specified as confidential. Hence, data obfuscation and de-obfuscation methods can perform in polynomial

time and scalable. In this approach, performing data obfuscation and de-obfuscation is the steps producing most overhead. Since the experimental results show that the data obfuscation and de-obfuscation do not cause much overhead, and hence my approach has reasonable performance.

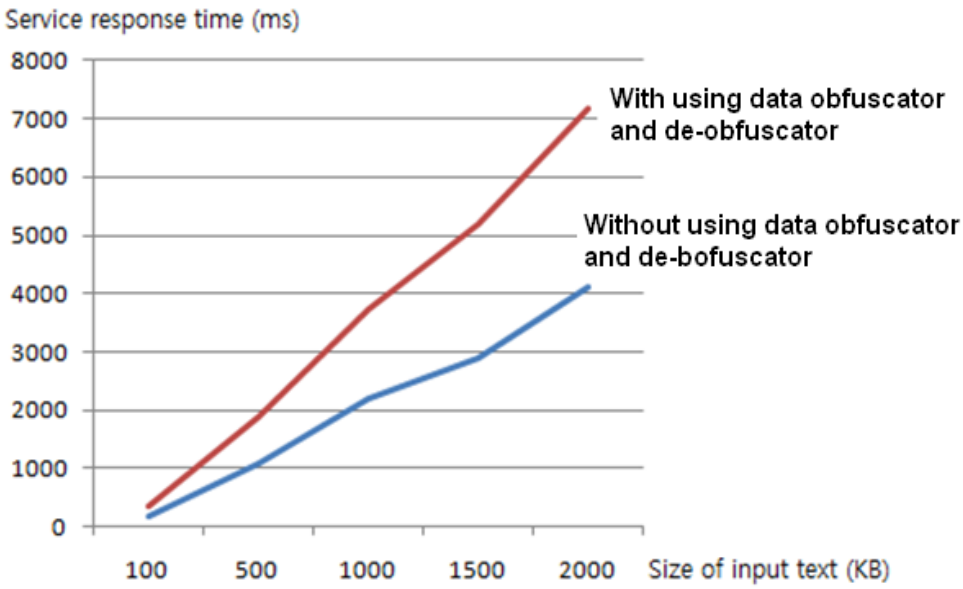

Figure 5: Service Response Time of Collaborative Online Documentation Service with Various Sizes of Input Texts

4.4 Security Evaluation of Data Obfuscation Methods

In this section, security evaluation of each data obfuscation method described in Section 4.1 is presented. The strengths of confidentiality protection provided by the obfuscation methods are measured based on two properties, difficulty of reverse-engineering the obfuscated data [60] and the resemblance of the obfuscated data with the original data [61].

- ***Evaluation of Method A)***

Given obfuscation function O, de-obfuscation function D, user's input data x, a function F and a secret integer key k, $O(x, k)$ generates dummy input data $a_1, a_2 \ldots a_n$, where $n > k$. For each dummy input data generated, the data obfuscator processes dummy functions $f_1(a_1)$, $f_2(a_2) \ldots f_n(a_n)$ while the software service instance processes $F(x)$. Then, an array of output data ($f_1(a_1)$, $f_2(a_2) \ldots f_{k-1}(a_{k-1})$, $F(x)$, $f_k(a_k), \ldots, f_n(a_n)$) is sent to the user. The data-de-obfuscator at the user's computer de-obfuscates the array of output data using the secret key k. Hence, $D((f_1(a_1), f_2(a_2) \ldots f_{k-1}(a_{k-1}), F(x), f_k(a_k), \ldots, f_n(a_n)) = F(x)$.

In this method, randomized noise to the original data provides non-resemblance of the obfuscated data. Let size of x is $S(x)$ and size of a_i is $S(a_i)$, where $1 \leq i \leq n$. Then the resemblance of the obfuscated data $\{x, a_1, a_2 \ldots a_n\}$ is measured by $\frac{S(x)}{S(x)+\sum S(ai)}$. The difficulty of reverse engineering increases when n increases. Let the time needed for reverse engineering function f_i is $T(f_i)$, where $1 \leq i \leq n$. Then the difficulty of the reverse engineering is measured by $\frac{T(F(x))+\sum T(fi(ai))}{T(F(x))}$. Users can adaptively increase the strength of confidentiality protection by increasing the variable n.

- ***Evaluation of Method B)***

  In this method, the deviation on file system provides non-resemblance of obfuscated data and difficulty on reverse engineering. Data obfuscator changes the number of disk sectors in a cluster and the format of the file allocation table in the file system so that the infrastructure service provider

cannot understand the file system structure. In order to prevent the service providers from extracting the users' specific confidential data from network traffics, the user's specific confidential data is encrypted in the user's computer before being sent to service provider's infrastructure.

- ***Evaluation of Method C)***

  In this method, the transformation of the format or value of users' input data provides non-resemblance of obfuscated data and difficulty on reverse engineering. All the operations associated with the original data must also be applicable to the transformed data so that the software service instance is able to process the obfuscated data. Given input data $\boldsymbol{x}$, a set of associated operations$\{\boldsymbol{P_1,\ldots, P_n}\}$, an obfuscation function $\boldsymbol{O}$, a secret key $\boldsymbol{k}$, and de-obfuscation function $\boldsymbol{D}$, $\boldsymbol{P_i(x) = D(P_i(O(x,k)),k)}$, where $1 \leq \boldsymbol{i} \leq \boldsymbol{n}$. Then Obfuscation function $\boldsymbol{O}$ and de-obfuscation function $\boldsymbol{D}$ have following properties:

  i) $\boldsymbol{D(F(O(x, k), k) = F(x)}$, where $\boldsymbol{k}$ is an obfuscation key unknown to the infrastructure service provider

  ii) The infrastructure service provider cannot understand the meaning of $\boldsymbol{x}$ by examining $\boldsymbol{O(x, k)}$

  iii) $\boldsymbol{D}$ or $\boldsymbol{k}$ cannot be derived from $\boldsymbol{O}$

Hence this method provides perfect non-resemblance of obfuscated data and makes the reverse engineering of the obfuscated data computationally impossible.

CHAPTER 5

ANONYMOUS SERVICE USAGE AND PAYMENT IN SERVICE-BASED SYSTEMS

This chapter presents an approach to protecting, from service providers, the anonymity of users' service usages and payment for their service usages in service-based systems. This approach has the following three features: 1) a user can access and use a service anonymously using his/her pseudonyms, 2) the user can pay the service provider for the his/her service usages without revealing his/her real identity, and 3) if the user performs improper activities, the user's pseudonyms are revoked and the user's real identity will be revealed by a trusted authority.

The anonymity discussed in this paper can be applicable at the transaction level between users and service providers. The "*anonymous service usage*" is defined as a user initiates a secure communication session with a service provider using a pseudonym, sends inputs to the service being used and receives outputs from the service. In addition, "*anonymous payment*" implies that a user makes payment for his/her service usages to the provider of the service without revealing his/her real identity.

Huang [62] presented a user identity anonymization approach in distributed computing systems, in which a user's real identity and pseudonyms are managed by separate trusted authorities. A user obtains a *certification ticket* from a trusted authority using his/her real identity, and then the user obtains his/her pseudonym certificates from another trusted authority using the *certification ticket*.

This architecture has a major advantage that a user's anonymity will not be compromised even if one of the trusted authorities is compromised by attackers.

In most cases of communications using Internet, the IP address of an entity is equivalent to its identity. Hence, it is important to hide the IP address in order to maintain the anonymity of its entity. A common way to hide the IP address is to use a proxy, which is a single server that accepts the identity of the connection from an *initiator* of an anonymous connection and forwards it to the *responder*, who is the service provider that the *initiator* wishes to contact anonymously [63, 64]. With this single-proxy method, the *responders* cannot see the actual IP addresses of the *initiators*. For example, the Anonymizer [63] and Lucent Personalized Web Assistant (LPWA) [64] use a proxy server to repackage users' web surfing requests to disguise their original IP addresses.

User-adaptive (or "personalized") applications aim at anticipating the needs of each individual user, and adapting to these needs while interacting with the user [65]. One problem of the user-adaptive applications is that user anonymization can be easily compromised in personalized systems because a user's activities are profiled and possibly linked to the real identity of the user. An approach for protecting user anonymity in the user-adaptive systems is presented in [66], in which users' anonymity is protected by pseudonyms and mix networks [67].

A distributed overlay network, called Tor [68], was developed to anonymize TCP-based communications over a computer network using onion routers [69]. Tor provides perfect forward security so that users can connect to

Internet sites without revealing their logical or physical locations to any sites on the Internet or to eavesdroppers. Tor can be easily used by many applications, like Web browsing, secure shell, and instant messaging. These approaches, however, are not suitable for SBS because they do not support anonymous payment of service usages.

A protocol for anonymous payment using anonymous digital cash was introduced in [70]. The anonymous digital cash is different from credit cards because the owner of the anonymous digital cash is non-traceable. This protocol works well for protecting the anonymity of the users' identities in any types of online transactions, but this protocol cannot be adopted in real world applications because this protocol can be easily abused by many criminals [71]. For example, a terrorist may threaten and force a bank to blindly sign digital cash without any personal contact with the bank. Then, the bank or police will have no way to trace the digital cash.

5.1 Anonymous Service Usage Protocol

An Anonymous Service Usage Protocol (ASUP) using public key infrastructure (PKI) certificates is developed, in which a user can use a service anonymously and pay for his/her service usages without revealing his/her real identity. In this paper, a protocol is defined as a formal description of digital message formats and the rules for exchanging those digital messages among communicating entities. ASUP involves the following four entities:

- A ***user*** who wishes to use a service anonymously.

- The ***Trusted Authority (TA)*** who is responsible for authenticating the users based on their real identities. After the users successfully pass the authentication of the TA, the TA issues a *pseudonym certificate* to each of the authenticated users for their anonymous service access.
- A ***Payment Authority (PA),*** who is a trusted entity and issues a *payment certificate* to a user, who requests to make a payment for his/her service usages. The *payment certificate* can be used as a proof to a service provider that the PA will pay for the user's service usages without revealing the user's real identity. The PA must have the information of the user's financial or billing information before issuing a *payment certificate* to the user. For example, a bank can issue a payment certificate for its customers who have sufficient deposit in its customers' accounts.
- A ***service provider (SP),*** who provides online services. A service provider may be malicious and interested in collecting and profiling users' online transactions.

In ASUP, it is assumed that each of the above four entities already has his/her own public and private key pair. ASUP uses the following three types of PKI certificates for anonymous service usage and payment:

- A ***real identity certificate (RICert),*** which is an electronic document containing a user's real identity information, such as name, address, or social security number, and the user's public key. The *RICert* is digitally signed by the TA's private key so that it is not forgeable or alterable.
- A ***pseudonym certificate (PCert)*** which is an electronic document containing a

user's pseudonym, the public key of the pseudonym and the user's *RICert*. The user's *RICert* is encrypted with the TA's private key so that no unauthorized entities can know the user's real identity from the *PCert*. The real identity of the user can be revealed by the TA only if a service provider detects the user conducting an illegal activity using the *PCert*. The *PCert* is digitally signed by the TA so that it is not forgeable or alterable.

- A ***payment certificate (PaCert)***, which is an electronic document issued to a user by a *PA*, which proves that the *PA* will pay for the user's service usage. A *PaCert* contains the *PA* name and *PCert* of the user. It is also digitally signed with the *PA*'s private key so that it is not forgeable or alterable.

Overall approach to anonymous service usage and payment is depicted in Figure 6, and can be summarized as follows:

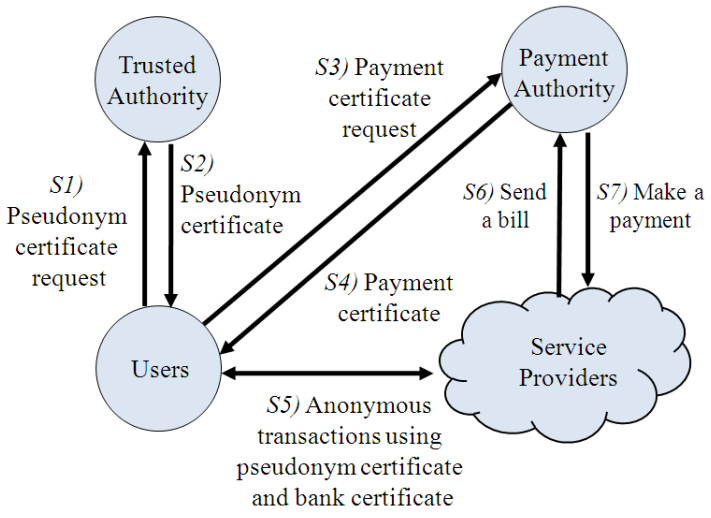

Figure 6: Anonymous Service Usage and Payment Using ASUP

An Approach for Anonymous Service Usage in SBS

S1) A user, who has a *RICert* issued by the TA, creates a pseudonym and requests the TA for a *PCert* of the pseudonym.

S2) The TA verifies the *RICert* of the user and issues a *PCert* to the user.

S3) The user requests a PA for a *PaCert*.

S4) The PA verifies *RICert* of the user and issues a *PaCert* for the user's pseudonym.

S5) The user uses a service anonymously using *PCert* and *PaCert*.

S6) The service provider sends a bill to the PA for the service usages by the user.

S7) The PA pays the bill.

The following types of possible attacks on ASUP are identified, which may attempt on the participants of the ASUP

A1) *Impersonation*. A malicious service provider or a third party attacker may steal a user's *PCert* or *PaCert* and attempt to impersonate the user.

A2) *Unauthorized bill*. A malicious service provider obtains a user's *PaCert*, and generates and sends a fake bill to the *PA*.

A3) *Multiple charges*. A service provider may charges the same service usages multiple times.

A4) *User profiling*. A service provider, whose services are used by a user, may profile the user based on the user's online transactions and attempt to link the user's pseudonyms to the user's real identity.

A5) *Malicious activities of a user*. A malicious user may use a fake *PCert* or

PaCert to perform illegal activities.

A6) ***Replay attack***. A third party attacker may launch replay attacks [72] to steal a user's *PCert* or *PaCert* certificates or waste system resources of the entities involved in ASUP.

A7) ***Refuse of payment***. A malicious user may refuse to pay the bill for his/her actual service usages because the service provider does not know the real identity of the user.

How ASUP can prevent all these types of attacks will be discussed in Section 5.3.

ASUP consists of the following four sub-protocols:

- Sub-protocol 1: The protocol for issuing *PCert*, which is described in Steps *S1)* and *S2).*
- Sub-protocol 2: The protocol for issuing *PaCert*, which is described in Steps *S3)* and *S4).*
- Sub-protocol 3: The protocol for anonymous service usage and billing, which is described in Step *S5).*
- Sub-protocol 4: The protocol for anonymous payment, which is described in Steps *S6)* and *S7).*

The following notations will be used in ASUP:

- ((P_t, S_t), (P_b, S_b), (P_{sp}, S_{sp}) and (P_a, S_a)) are the public/private key pairs of TA, PA, service provider and user, respectively.

- $Time_t$, $Time_b$, $Time_{sp}$ and $Time_a$ are the time stamps generated by TA, PA, service provider and user, respectively.
- H is the hash function used in ASUP.
- $E_k(M)$ is an encrypted message M using the encryption key K

It is assumed that all the message exchanges in ASUP occur through insecure communication networks, such as Internet, and all the messages can be eavesdropped by malicious third parties. The description of each of these four sub-protocols is as follows:

Description of *Sub-protocol 1*:

1-1) The user has a *RICert* = [ID_a, $E_{St}(H(ID_a)$], where ID_a = [Name, Address, SSN, P_a]

1-2) The user sends a request $RQ_1 = [E_{Pt}(req_p, RICert, Time_a)]$ to the TA, where req_p indicates that this message is for requesting a *PCert*.

1-3) Upon receiving RQ_1, the TA decrypts RQ_1 and verifies the *RICert* using P_t. The TA checks its database to ensure that the *RICert* is neither expired, nor revoked.

1-4) Upon successful verification of *RICert*, the TA generates a random variable r_1, and sends $CH_1 = [E_{Pa}(req_p, r_1, Time_t)]$ to the user.

1-5) Upon receiving CH_1, the user decrypts CH_1, creates a pseudonym *PS* and sends $RP_1 = [E_{Pt}(req_p, r_1, PS, Time_a)]$ to the TA.

1-6) Upon receiving RP_1, the TA decrypts RP_1, verifies r_1, and issues a *PCert* for the user. The contents of the *PCert* are as follows:

$PCert$ = [TA, PS, $Time_t$, $H(RICert \| r_1)$, $E_{Pt}(RICert \| r_1)$, Sig_{TA}], where || is a concatenation operator, and

$Sig_{TA} = E_{St}(H(\text{TA} \| PS \| Time_t \| H(RICert \| r_1) \| E_{Pt}(RICert \| r_1))$

1-7) The TA sends $E_{Pa}(PCert)$ to the user.

1-8) The user decrypts $E_{Pa}(PCert)$ and obtains the *PCert*.

1-9) The TA maintains the list of *PCerts* issued for each user in its database. When an illegal activity of a user is reported, TA revokes the *RICert* and all the *PCerts* associated with the user.

Description of *Sub-protocol 2*:

2-1) The user sends a request $RQ_2 = [E_{Pb}(req_b, RICert, r_1, PCert, Time_a)]$ to a PA, where req_b is a pre-defined value indicating that this message is for the *PaCert* issuing process.

2-2) Upon receiving RQ_2, the PA verifies *RICert* using P_t, calculate $H(RICert \| r_1)$ and verifies whether the *PCert* is issued for the *RICert*.

2-3) Upon successful verification of *RICert* and *PCert*, the PA generates a random variable r_2, and sends $CH_2 = [E_{Pa}(req_b, r_2, Time_b)]$ to the user.

2-4) Upon receiving CH_2, the user decrypts CH_2, generates a public/private key pair (P_{pseu}, S_{pseu}) and sends $RP_2 = [E_{Pb}(req_b, r_2, P_{pseu}, Time_a)]$ to the PA

2-5) Upon receiving RP_2, the PA decrypts it, verifies the r_2, generates an agreement AG = [PA name, *PCert*, the user's a*ccount#*, P_{pseu}], and sends $E_{Pa}(AG)$ to the user.

2-6) Upon receiving *AG*, the user verifies the *AG*, and sends E_{Pb} (*SignedAG*) to the PA, where *SignedAG* = [E_{Sa}(*H*(*AG*))]

2-7) Upon receiving *SignedAG*, the PA verifies the *SignedAG* using P_a, and stores the agreement (*AG* || *SignedAG*) in the PA's database.

2-8) The PA issues a *PaCert* for the user with the following contents :

PaCert = [PA name, *PCert*, P_{pseu}, $Time_b$, expiration date, Sig_b], where

$Sig_b = E_{Sb}$(*H*(PA name || *PCert* || P_{pseu} || $Time_b$ || expiration date))

2-9) The PA sends E_{Pa}(*PaCert*) to the user.

2-10) The user decrypts E_{Pa}(*PaCert*) and obtains the *PaCert*.

Description of *Sub-protocol 3*:

3-1) The user sends *PCert* and *PaCert* to a service provider

3-2) The service provider verifies the authenticity of the certifications using P_t and P_b.

3-3) The service provider verifies with the TA whether the the *PCert* has been revoked.

3-4) If the *PCert* is valid, the service provider generates a session key *K* and sends $Ini_1 = E_{Ppseu}$(*K*, expiration time of *K*, E_{Ssp}(*H*(*K*)) to the user.

3-5) Receiving on Ini_1, the user decrypts it, verifies *H*(*K*) using P_{sp}, and starts a secure communication session with the service provider using *K*. Note that *K* is a symmetric key shared between the user and service provider .

3-6) The user sends *SR* =[*service_request*, $Time_a$, E_{Spseu}(*H*(*service_request* || $Time_a$)], where *service_ request* =[*PCert*, service name, user inputs]

3-7) The service provider stores *SR* in its database. The service provider also maintains audit trails for processing the service request of the user.

3-8) After completing the process of the service request, the service provider send $E_{Ssp}(Bill)$ to the user, where *Bill* = [*service_request*, $amount, $Time_{sp}$].

3-9) Upon receiving $E_{Ssp}(Bill)$, the user decrypts it using P_{sp}, verifies the amount, generates a random unique string *U*, and sends *SignedBill* = [*Bill*, *YCert*, *U*, $E_{Spseu}(H(Bill \parallel YCert \parallel U))$]. Note that *U* should be long enough to reduce the chance of another person also using it to a negligible level.

Description of *Sub-protocol 4*:

4-1) The service provider sends $B = E_{Ssp}(SignedBill, Time_{sp})$ to the PA.

4-2) Upon receiving B, the PA decrypts it using P_{sp}, verifies the signature of *SignedBill* using P_{pseu}.

4-3) The PA checks its database to ensure that a *SignedBill* with the same unique string *U* has not been previously paid.

4-4) If the *SignedBill* has not been paid before, the PA accepts the bill and makes a payment to the service provider.

4-5) The PA stores *SignedBill* in its database.

The message exchanges among entities in each step of the ASUP are summarized in Table 3.

Table 3: Message Exchanges in ASUP

S1)	user →TA: $RQ_1 = [E_{Pt}(req_p, RICert, Time_a)]$
	TA→user: $CH_1 = [E_{Pa}(req_p, r_1, Time_t)]$
	user →TA: $RP_1 = [E_{Pt}(req_p, r_1, PS, Time_a)]$
S2)	TA→user: *PCert* = [TA, *PS*, $Time_t$, $H(RICert \| r_1)$, $E_{Pt}(RICert \| r_1)$, Sig_{TA}]
S3)	user →PA: $RQ_2 = [E_{Pb}(req_b, RICert, r_1, PCert, Time_a)]$
	PA→user: $CH_2 = [E_{Pa}(req_b, r_2, Time_b)]$
	user→PA: $RP_2 = [E_{Pb}(req_b, r_2, P_{pseu}, Time_a)]$
	PA→user: *AG* =[PA name, *PCert*, the user's a*ccount#*, P_{pseu}])
S4)	user→PA: *SignedAG*
	PA→user: PaCert = [PA name, *PCert*, P_{pseu}, $Time_b$, expiration date, Sig_b]
	user→SP: *PCert* ‖*PaCert*
	SP→user: $Ini_1 = E_{Ppseu}(K$, expiration time of K, $E_{Ssp}(H(K))$ user→SP: *SR* =[*service_request*, $Time_a$, $E_{Spseu}(H($*service_request* ‖ $Time_a)]$
	SP→user: *Bill* = [*service_request*, $amount, $Time_{sp}$] user→SP: *SignedBill* = [*Bill*, *PaCert*, *U*, $E_{Spseu}(H($*Bill* ‖ *PaCert* ‖ *U*$))]$
S5)	SP→PA: $B = E_{Ssp}(SignedBill, Time_{sp})$
S6)	PA →SP: make a payment

5.2 Anonymity Assessment

The criteria for evaluating anonymity of a user in a system are presented in [73-74]. I will discuss each of these criteria and show that ASUP satisfies each of these criteria.

- ***Sender-anonymity*** means that the identity of the party who sends a message is

hidden. In my approach, a user initiates a service request to a service provider. Hence, the user is considered as the *sender*. The sender's anonymity of the sender can be achieved by the sender using pseudonyms. The pseudonyms are generated by the user so that service providers cannot infer the real identity of the user from the pseudonyms. For the billing and payment processes, the user uses a *PaCert* to make an anonymous payment without revealing his/her real identity so that the sender anonymity can still be protected.

- ***Receiver-anonymity*** means that the identity of the party who receives a message is hidden. ASUP is mainly designed for anonymous use of SBS. Since the service providers of a SBS are usually publically known, the receiver anonymity in my approach is not considered.
- ***Unlinkability*** means that a user may make multiple transactions with a service without the service provider being able to link these transactions together. The unlinkability ensures that the users' online transactions cannot be profiled by service providers. In ASUP, a user can obtain *PCert*s as much as he/she wants, as long as he/she does not violate the sub-protocol 1 with TA. This means that the user may use a new pseudonym for each transaction with a service. In this way, the unlinkability can be achieved based on the following assumption: SBS usually involves a large number of anonymous users, and the service providers will not be able to link different pseudonyms to the same user by observing the transactions.
- ***Unobservability*** means that certain items of interest (IOI), such as message, event, and transaction, are not distinguishable from any other IOI. This means

that third-party adversaries are not able to discern user's messages from a random noise. ASUP provides unobservability for all the messages exchanged among entities because the messages are encrypted before being sent out through the Internet.

- ***Managability*** means that if a user performs an illegal activity using a pseudonym, all the pseudonyms associated with the user are revoked and the real identity of the user can be traced. The revocation of all the pseudonyms of the user should prevent the user from accessing any service. For example, if a user has ten pseudonyms and performs an illegal activity with one of the pseudonyms, then all ten pseudonyms of the user must be revoked to prevent the user from further communicating with other service providers. In addition, revoking the real identity of the user is necessary to prevent the user from obtaining new pseudonyms. In ASUP, the TA maintains the list of *PCerts* issued for each user in its database. When an illegal activity of a user is reported, TA revokes the *RICert* and all the *PCerts* associated with the user. How a service provider reports an illegal activity of user will be discussed in Subsection 5.3. Every service provider verifies with the TA whether a user's pseudonym has been revoked before establishing a communication session with the user so that revoked users cannot access any service.
- ***Admissibility*** means that a digital document can be submitted to the legal authority, such as the court as an evidence of transaction or an incident. The admissibility of ASUP is provided by the digital signature scheme using PKI. Since any of digitally signed documents is not alterable nor forgeable, the

following digital documents used in ASUP are admissible: 1) a *PCert* signed by the TA, which proves the real identity of the owner of the *PCert*, 2) a *PaCert* signed by a PA, which proves the real identity of the owner of the *PaCert*, 3) a *service_request* signed by a user, which proves that the user sent the *service_request* to the service provider, and 4) a bill signed by a user, which proves that the user approved to pay for the bill.

5.3 Security Assessment

This section will discuss how ASUP prevents the possible attacks identified in Section 5.1.

- ***Impersonation*:** ASUP provides unobservability for all the messages exchanged among entities. Hence, a user's *PCert* or *PaCert* cannot be stolen by third-party adversaries through eavesdropping the communications among entities in ASUP. A user's *PCert* or *PaCert* can be obtained by a malicious service provider, who may later attempt to impersonate the user. In order to prevent a malicious service provider from impersonating a user, ASUP uses the challenge-response mechanisms in *Steps 1-4) and 1-5)* of *Sub-protocol 1*, *Steps 2-4) and 2-5)* of *Sub-protocol 2*, and *Steps 3-4) and 3-5)* of *Sub-protocol 3*. Hence no service provider can impersonate a user without knowing the user's private keys, which are only known to the user.
- ***Unauthorized bill***: In order to prevent a malicious service provider from faking a bill using a user's *PaCert*, ASUP requires service providers to get the user's digital signature before sending a bill to a PA in *Step 3-9)* of *Sub-protocol 3*. If a

service provider sends a bogus bill to a PA, the PA will notice it by verifying the user's digital signature.

- ***Multiple charges****:* In order to prevent a malicious service provider from sending the same bill to a PA more than once, ASUP requires a user to generate a random unique string and attach it to a bill before signing the bill in *Step 3-9)* of *Sub-protocol 3*. Every time a PA pays for a bill, the PA verifies the unique number attached to the bill by checking whether the bill with the same unique string has been paid before. After successful verification and payment of the bill, the PA stores the unique number attached to the bill in its database. Service providers cannot modify the unique string on the bill because it is digitally signed by the user.
- ***Malicious activities of user****:* As discussed in Subsection 5.1, ASUP provides the manageability for revoking the *PCerts* and *RICert* of a user who performed illegal activities. Service providers maintain audit trails for users' activities in their services in *Step 3-7)* of *Sub-protocol 3*. When improper activities of a user are detected, the TA reveals the real identity of the user from the user's *PCert*, and revokes all the *Pcerts* associated with the user from its database. *Step 1-3)* in *Sub-protocol 1* and *Step 3-3)* in *Sub-protocol 3* ensure that a revoked user cannot receive a new *PCert* from the TA or access a service using previously issued *PCerts*.
- ***User Profiling***: As discussed in Section 5.1, ASUP provides unlinkability through allowing a user has multiple *PCerts*.
- ***Replay Attack***: In ASUP, all the messages contain time stamps, denoted by

$Time_t$, $Time_b$, $Time_{sp}$ and $Time_a$. The timestamps prevent the adversaries from launching replay attacks. Any message replayed by an attacker can be detected through the verification of the timestamps [72].

- ***Refuse of payment****:* ASUP requires users to sign *service-requests* before they send their *service-requests*. As discussed in Section 5.1, a *service_request* signed by a user proves that the user sent the *service_request* to the service provider, and it is non-repudiative and admissible.

CHAPTER 6

SOFTWARE MODULE DECOMPOSITION AND DISTRIBUTED EXECUTIONS IN SBS

This chapter presents an approach to software module decomposition and distributed executions in service-based systems to protect users' data confidentiality.

6.1 Overview of SBS Application Development

The basic underlying idea of the approach to protecting users' confidential data is that a SBS is divided into software providers and infrastructure service providers as shown in Figure 7. Then an application is decomposed into smaller software modules, and those software modules are distributed to different infrastructure service providers for executions. By decomposing application into software modules and distributing their executions, no infrastructure service provider knows overall functionality of the software nor has the entire set of users' data.

In the approach to software decomposition and distributed execution, users first provide data specifications to application developers. The data specification specifies users' data sets and their requirements on data confidentiality. The confidentiality requirements define rules for) how the data should be stored in SBS 2) how the data should be flowed in SBS and 3) how the data should be processed in SBS. The data specifications can be specified using a data specification language, which will be discussed in Section 6.2. SBS application

providers develop SBS applications based on the data specifications provided by users. A SBS application consists of multiple software modules, which will be distributed and executed in distinct infrastructures of data processing service providers. The SBS application satisfies the users' data confidentiality requirements using data obfuscation, anonymization of users' identities, partitioning users' data set and controlling the data flows in SBS. The software module decomposition process consists of two phases: requirement phase and development phase.

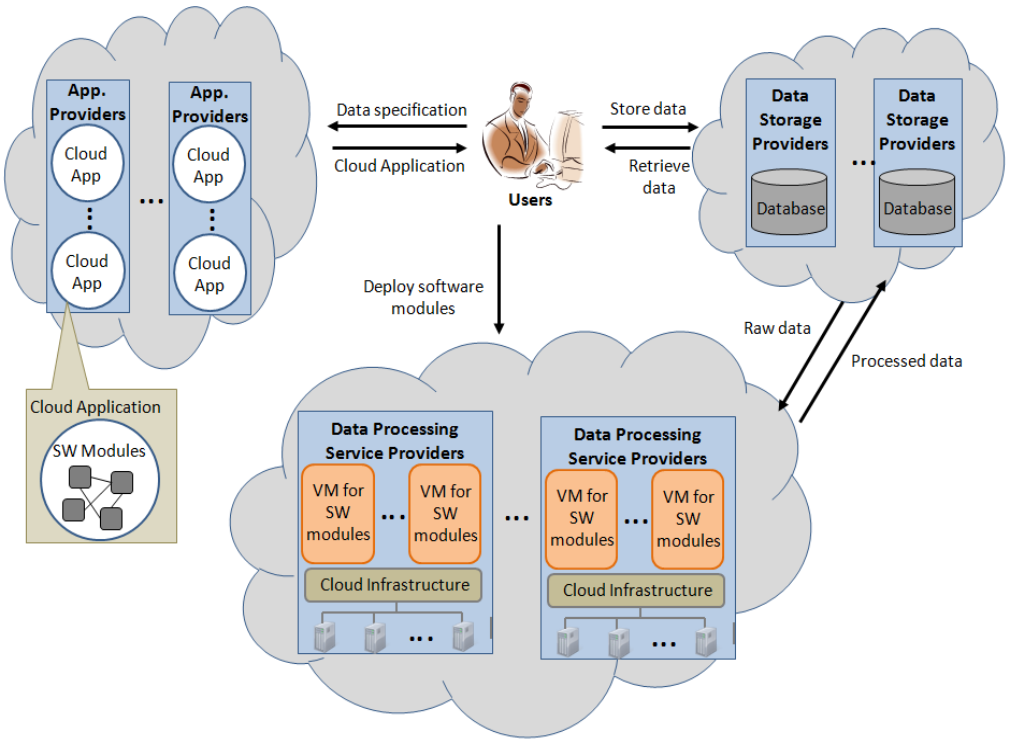

Figure 7: An Approach to Software Decomposition and Distributed Execution

6.2 Requirement Phase

In the requirement phase, a user specifies his/her data sets and requirements on data confidentiality using a data specification model. A data specification model is developed, which has the following seven entities as shown in Figure 8:

- ***Data Set*** consists of Data Description, Data Classification, Personally Identifiable Information, Confidentiality Requirements, Encryption Key and Obfuscation Key. User's entire data is partitioned into multiple data sets.

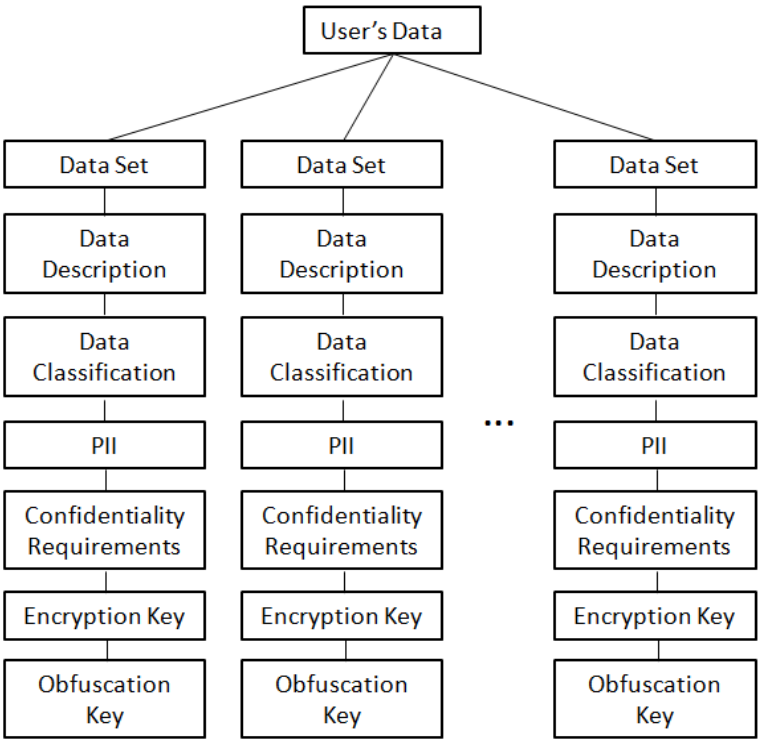

Figure 8: The Format of Data Specification of Users' Confidentiality Requirements

- ***Data Description*** describes data type and data structure in the data set. The data can be structured data or unstructured data. Structured data includes data

tables, data columns and data rows in relational database, and data trees in XML database. Unstructured data includes document files, images and videos.

- ***Data Classification*** specifies sensitivity of the data. The data in the data set can be classified in three levels as shown in Table 4.
- ***Personally Identifiable Information (PII)*** specifies PII in the data set, such as names, addresses and social security numbers.
- ***Confidentiality Requirements*** specifies 1) how the data should be stored in SBS, 2) how the data should be processed in SBS and 3) how the data should be flowed in SBS. For the data storage, user specifies the data set to be stored in the infrastructure in a plain form, an obfuscated form or an encrypted form. For the data processing, user specifies the data set to be processed in the infrastructure of service providers in a plain form or an obfuscated form. If the user wants the data set to be processed using data obfuscation, he/her also specifies which obfuscation method to be used. For the data flow in SBS, user specifies data sets that should not be processed by same data processing service providers. It is defined that data sets ***D1*** and ***D2*** is *incompatible* if ***D1*** and ***D2*** should not be processed by the same infrastructure provider.
- ***Encryption Key*** to be used to encrypt the data set.
- ***Obfuscation Key*** to be used to obfuscate the data set.

Table 4: Data Classification to Specify Sensitivities of Users' Data

	Public	Private	Confidential
Description	Not sensitive; available to anyone	Slightly sensitive; include personal information	Extremely sensitive; should not disclosed to any of unauthorized entities
Impact of unauthorized disclosure	N/A	Violation of privacy of individual person	Data confidentiality is compromised
How to store	Stored in a plain format	Stored in an encrypted or obfuscated format	Stored in an encrypted format
How to process	Processed in a plain format	Processed using data obfuscation or user identity anonymization. If data obfuscation and user identity anoymization is not applicable, must be processed in a trusted domain.	Processed using data obfuscation. If data obfuscation is not applicable, must be processed in a trusted domain.

A specification language for users to specify their data confidentiality requirements is developed. The syntax of the specification language is described in Backus-Naur Form (BNF) as follows:

<User's Data> ::= <Data Set>$^{+}$

<Data Set>:: = "(" {<Set ID#>, <Data Description>, <Data Classification>, <Personally Identifiable Information>, <Confidentiality Requirements>, <Encryption Key>, <Obfuscation Key>} ")"

<Data Description> ::= "(" {<Unstructured Data>* , <Structured Data>*} ")"

<Unstructured Data> ::= <Documents>* <Images>* <Videos>* <Others>*

<Structured Data> ::= <Data tables>* <Data columns>* <Data rows>* <XML trees>*

<Data Classification> ::= "Public" | "Private" | "Confidential"

<Confidentiality Requirements> ::= "(" { <Storage Requirements>, <Process Requirements> , <Data Flow Requirements> } ")"

<Storage Requirements> ::= "Plain" | "Encrypted" | "Obfuscated"

<Process Requirements> ::= "Plain" | <Obfuscation Method>

<Obfuscation Method> ::= "Method1" | "Method2" | "Method3"

<Data Flow Requirements> ::= "({<Set ID#>*} ")"

In the requirement phase, Function-Oriented Design Model [75] is used to identify the functional units of the application, and Data-Flow Diagram (DFD) [76] is used to identify data sets associated with the inputs of each of the functional units. Then a user specifies his/her data confidentiality requirements on each data set using the specification language. The user can generate his/her data confidentiality requirements as follows:

Confidentiality Requirements Specification Process

Step 1) Application developer identifies the functional units of the application using the Function-Oriented Design Model

Step 2) The application developers construct Data-Flow Diagram (DFD) to show data flows among functional units.

Step3) The user identifies his/her data sets associated with the inputs of each functional unit.

Step 4) The user specifies his/her confidentiality requirements by the following procedure:

1 S = {s | s is data sets identified in *Step3*}

2 For each data set $s_1 \in S$:

3 Specify <Data Description> of s_1 by choosing <Documents>, <Images>, <Videos>, <Data tables>, <Data columns>, <Data rows> or <XML trees>

4 Specify <Data Classification> of s_1 from Table 4.

5 If s_1 must be stored in a encrypted form,

Set <Storage Requirements> of s_1 = "Encrypted"

6 If s_1 must be stored in a obfuscated form,

Set <Storage Requirement> of s_1 = "Obfuscated"

7 If s_1 must be processed in a obfuscated form,

Specify which obfuscation method to be used for s_1

8 For each data set $s_2 \in S$:

If s_1 and s_2 are incompatible,

To show this process, consider a credit card company wants to develop an application to automatically generate bills for its customers. The company has two data sets, which are two columns: Name and Purchase History. The column Name contains names of the company's customers, and the column History contains the purchasing histories of each customer. The company wants to make sure that

Infrastructure providers of SBS cannot link a customer with any purchasing history. The user, who is the company, specifies its data confidentiality requirements by following the approach presented in Section 6.2.

Step 1) Three functional units of the application are identified as follows:

Function F1: Sort the column Name by customers' last name

Function F2: Find all purchasing history for each customers

Function F3: Find total amount of purchases for each customers

Step 2) Customers' name and purchasing histories are both classified as confidential.

Step3) Associate the columns Name and Purchase History to Functions F1 and F2 respectively.

Step 4) Using the results in the above steps and the specification procedure discussed in Section 6.2, the confidentiality requirements are:

<User's Data> ::= <S_1> <S_2>

<S_1>:: = "(" {<1>, <Data Description1>, <Data Classification1>, <Confidentiality Requirements1>, <Encryption Key1>, <Obfuscation Key1>} ")"

<S_2>:: = "(" {<2>, <Data Description2>, <Data Classification2>, <Confidentiality Requirements2>, <Encryption Key2>, <Obfuscation Key2>} ")"

<Data Description1> ::= column Name

<Data Description2> ::= column History

<Data Classification1> ::= "Confidential"

<Data Classification2> ::= "Confidential"

<Confidentiality Requirements1> ::= "(" { <Storage Requirements1>, <Process Requirements1> , <Data Flow Requirements1> } ")"

<Confidentiality Requirements2> ::= "(" { <Storage Requirements2>, <Process Requirements2> , <Data Flow Requirements2> } ")"

<Storage Requirements1> ::= "Encrypted"

<Process Requirements1> ::= "Method3"

<Storage Requirements2> ::= "Encrypted"

<Process Requirements2> ::= "Method3"

<Data Flow Requirement1> ::= 2

<Data Flow Requirement2> ::= 1

6.3 Application Development Phase

In the development phase, application developer develops a SBS application based on user's data specification and confidentiality requirements. A SBS application consists of multiple software modules, which will be distributed and executed in distinct infrastructures of the SBS service providers.

How to achieve high cohesion of individual software modules and low coupling among software modules are major challenges of developing distributed and parallel applications. High cohesion of a software module means that there should be many interactions among the functions of the module, and the module developers must provide well-defined interfaces for using the functions of the module by other modules. High **c**oupling among modules means that the execution of one module depends on the implementation of other modules, and it causes not only a burden for code evolution of the module, but also for reusing modules in a different application context. It also makes distributed deployment and maintenance very complex and unmanageable for large SBSs because debugging and testing of an application with high coupling among modules is very time-intensive and expensive. In order to achieve low coupling among modules, standard sets of interfaces and protocols among distributed modules in the SBSs are needed so that module developers are able to implement the functionalities of modules independently without concerning with the implementation of other modules. In addition, standard platforms for the deploying and executing modules are needed. These platforms automatically manage the deployment and execution of the modules so that module developers do not need to worry about the complexity of module deployment and execution environments during the module development phase. In the remainder of this section, an approach to achieve high cohesion of individual modules and low coupling among modules will be presented.

In [77], I have presented a software engineering framework for combining

these services computing and cloud computing paradigms in a software engineering framework to achieve high cohesion of individual modules and low coupling among modules. The reasons that I combined these two computing paradigms are as follows. Services computing and cloud computing are two separate paradigms, and each has certain advantages for application development. Application developers can use services computing alone, cloud computing alone, or a combination of the two. However, using only one of the computing paradigms alone cannot achieve both high cohesion of individual module and low couplings among the modules. For example, a major challenge of developing SBSs using SOA is lack of methods for deployment of software modules on distributed servers. As a consequence, there may be high couplings between modules and platforms for executing the modules. Cloud computing can help meet that challenge through the use of standardized platforms and virtualization of computing resources. On the other hand, there are few standard sets of interfaces and protocols among distributed software modules for cloud computing systems. Consequently, high cohesion of individual modules and low coupling among modules cannot be achieved by cloud computing alone. Services computing can help meet that challenge through service oriented architecture (SOA) standard interfaces for software modules and communication protocols among the software modules. The standard sets of interfaces among distributed modules help module developers implement the functionalities of modules independently without concerning with implementations of other modules. The standard sets of protocols among distributed modules help module developers

provide well-defined and highly cohesive functionalities of the module. Figure 9 shows tour software engineering framework for module development.

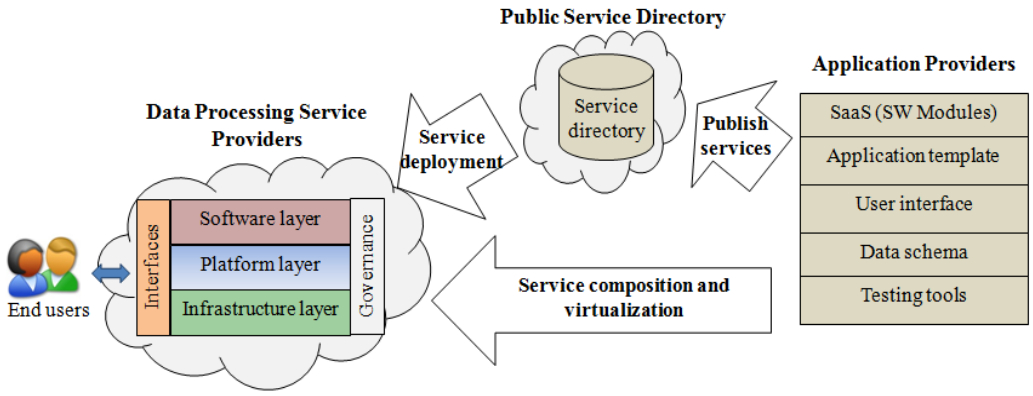

Figure 9: Software Engineering Framework for Combining Services Computing and Cloud Computing to Achieve High Cohesion and Low Coupling among Software Modules

The module developers follow standard interfaces of SOA when they develop software modules. The functionalities of software modules are delivered through the standard protocols of SOA. Standard platforms for the deploying and executing modules are provided by cloud computing virtualization technology.

In my framework, application developers do not need to concern with the complexity of module deployment and execution environments in the application development phase because any computing environments and resources required for executing the software modules, such as operating systems, platforms, communication channels and system resources, can be dynamically virtualized in the SBS infrastructures of infrastructure service providers. Each software module can be developed by the software module developers using the software

development methods available for centralized and monolithic applications, such as object-oriented programming and aspect-oriented programming. Most importantly, each module can be tested in isolation and on a single machine, using agile and iterative development techniques, which are generally much harder to use for distributed applications.

In my framework, the decomposition of an SBS application into software modules to satisfy users' data confidentiality requirements can be done as follows:

Software Decomposition Process

Step 1) Identify the functional units of the application based on application requirements using Function-Oriented Design Model [75]

Step 2) Construct Data-Flow Diagram (DFD) [76] to show data flows among functional units.

Step3) Identify users' data sets associated with each functional unit

Step 4) Transform the Data-Flow Diagram to undirected weighted tree. Each node represents a functional unit. Each edge represents data flows between nodes. The weight of each edge represents the number of data flows between nodes in the Data-Flow Diagram.

Step 5) Find the optimal software decomposition using the following Greedy Algorithm:

T = Weighed tree

N = {n | n is a node in T}

E = {e | e is a edge in T}

E' = {e | e connects nodes n1 and n2 ∈ N, and n1 and n2 are compatible}

M = {m | m is a software module}

While (E' ≠ Ø)

Find e ∈ E' which has the biggest weight

Integrate n1 and n2 into software module M

Remove n1 and n2 from N

Remove e from E

Remove e from E'

E' = {e | e connects nodes n1 and n2 ∈ N, and n1 and n2 are compatible}

To illustrate this process, consider the same example of the credit card company who wants to develop an application to automatically generate bills for its customers in Section 6.2. The company has two data sets, which are two columns: Name and Purchase History. The two columns are specified as confidential data by the company.

Step 1) The same step with the *Step1)* of the Confidentiality Requirements Specification Process in Section 6.2, and the functional units are:

Function F1: Sort the column Name by customers' last name

Function F2: Find all purchasing history for each customers

Function F3: Find total amount of purchases for each customers

Step 2) The generated data-flow diagram (DFD) is given below

F = {F1, F2, F3}

Data Flows = {F1 → F2, F2 → F3}

Step3) The same step with the *Step3)* of the Confidentiality Requirements Specification Process in Section 6.2, and the columns Name and Purchase History are associated to Functions F1 and F2 respectively.

Step 4) Transform the Data-Flow Diagram to the following undirected weighted tree:

T = Weighed tree

N = {F1, F2, F3}

Step 5) Using the results in the above steps and the Greedy algorithm, the module decomposition are:

Software Module = {M1, M2}

F1 ∈ M1, F2 ∈ M2, F3 ∈ M2

CHAPTER 7

ADAPTIVE RESOURCE ALLOCATION FOR SBS

In this Chapter, an approach to adaptive resource allocation for SBSs will be presented.

7.1 Background for Adaptive Resource Allocation in SBS

The rapid growth of the Internet has enabled many applications involving many users and service providers distributed in various geographical locations. In order to facilitate the development of such applications, service-oriented architecture (SOA) has been adopted in various distributed systems, such as web service, grid computing systems, utility computing systems and SBS computing systems. Such systems are referred as service-based systems (SBS), and leads to the vision of "Internet as a supercomputer." This vision incorporates the concepts of "software as a service", "platform as a service", "infrastructure as a service" and "resources as a service," and requires software systems running on the open and dynamic Internet with the capabilities of context-awareness, self-management and autonomous adaptation for changes in runtime environment. Such systems residing on the Internet are referred in general as Internetware systems.

A major challenge for the Internetware systems to satisfy various application requirements is to manage the quality of service (QoS) in runtime due to the dynamic, loosely coupled, and compositional nature of SOA. Many studies have identified important QoS features of SOA systems [78, 79], such as throughput, timeliness and security, which are directly affected by the limitation

of system resources. It is noted that the system resource considered here is for large-scale aggregation of distributed computing resources working together over the Internet as a tremendous virtual computer [80-82]. For example, computing grids and SBSs are vast resource pools for on-demand requests over the Internet. In order to manage the QoS for such systems, an effective resource allocation technique for dynamically scalable distributed system resources is needed. In order to achieve this goal, the SBS needs the capabilities of monitoring the changing system status, analyzing and controlling system QoS features, and adapting its service configuration to satisfy the QoS requirements of multiple workflows simultaneously. Since multiple services are often hosted by the same server in SBS, and the services in the same server compete for the limited available resources of the server, such as CPU-time, memory and network bandwidth, different resource allocations will result in different QoS in runtime. In addition, the service compositions occur dynamically in runtime and with the resource status of servers dynamically changing. Thus, allocating the resources of each server to its services for successfully satisfying the QoS requirements of dynamic multiple workflows and resource status is needed to satisfy the overall QoS requirements of the workflows in SBS. In practice, a service may create multiple service instances working for different workflows [83]. Two different service instances are considered as two different services. Although the functionalities of the multiple service instances are exactly the same, the amount of resource consumed by the service instances may vary depending on the

workflows and services they deployed in their servers. Hence, I will consider different service instances as different services.

This section will present the challenges of adaptive resource allocation in SBS, and the current state of the art in handling dynamic resource allocation for various computing and network systems which are useful for dynamic resource allocation in SBS. I will then present an approach to adaptively allocating the system resources of servers to their services in runtime to satisfy one of the most important QoS requirements, the throughput, of multiple workflows in SBS.

Development of QoS and resource management middleware has been studied in various areas, such as real-time systems, distributed object-oriented systems, web services, grid computing and SBS computing. A workflow-based computational resource broker [84, 85] was presented for grid computing environment. The main function of the resource broker is to monitor the available resources and match workflow requirements to available resources over multiple administrative domains. The resource broker provides a uniform interface for accessing available system resources of computing grid via users' credentials. An open-source toolkit Globus has emerged as a standard middleware for resource management in grid computing environment [86]. Globus provides Web Service Grid Resource Allocation and Management (WS GRAM) Protocol in which a set of web services are designed to support APIs for requesting and using grid system resources. Globus also provides Monitoring and Discovery Service (MDS) that supports a common interface for collecting contextual information on system

resources, such as available processors, CPU load, network bandwidth, file system information, storage devices, and memory in computing grids.

A market-based autonomic resource management approach in SBS computing environment was developed [87], in which Service-Level Agreement Resource Allocator provides the interfaces between the SBS service providers and external users/brokers for 1) monitoring users' service requests and QoS requirements, 2) monitoring the availability of system resources, 3) examining service requests to determine whether to accept the requests according to resource availability and processing workload, 4) allocating system resources to virtual machines in SBS, 5) pricing the usage of resources and prioritizing the resource allocation, and 6) keeping track of the execution progress of the service requests and maintaining the actual usage of resources. This approach supports negotiation of QoS between users and providers to establish service-level agreements (SLA) and allocation of system resources to meet the SLAs.

Multi-layered resource management (MLRM) architecture using standard-based middleware technology [88] was developed for enterprise distributed real-time embedded (DRE) systems. This architecture supports dynamic resource management to optimize and reconfigure system resources at runtime in response to changing mission needs and resource status. With the dynamic resource allocation, a DRE system can provide QoS for critical operations under the overloaded and resource constrained environments. In [89], a constraint-programming-based approach was presented to solve resource allocation problem in real-time systems. The problem of assigning a set of preemptive real-time tasks

in a distributed system is formulated as a constraint satisfaction problem (CSP) with allocation, resource and timing constraints. First, the CSP is solved using constraint programming techniques to satisfy the allocation and resource constraints. Then, the solution is validated for timing constraints through Logic-based Benders decomposition [90].

In [91], a decentralized local greedy mechanism for dynamic resource allocation in web service applications was presented. In this mechanism, software agents are generated to buy and sell network services and resources to and from each other. In [92], a market based resource allocation for web services in a commercial environment was presented. In this approach, service providers employ a cluster of servers to host web services, and service consumers pay for service usage with QoS requirements, such as waiting time or response time. A framework was developed for evaluating the effect of particular resource allocation in terms of performance and average revenue earned per unit time. A heuristic algorithm was also developed for making resource allocation decisions in order to maximize revenue.

The QoS of networks has been investigated for providing prioritized services by efficient resource allocation techniques through labeling, scheduling and routing mechanisms. In [93], an optimal resource allocation and pricing scheme for next generation multiclass networks was presented. In this scheme, an optimization problem is formulated based on a nonlinear pricing model, whose solution ensures satisfaction of network delay constraints and efficient resource allocation in dynamic multiservice networks. In [94], an energy-efficient radio

resource allocation approach based on game theory for wireless networks was presented. The study shows that the game-theoretic approach for resource allocation is useful for energy-constrained wireless networks. In [95], a resource allocation scheme for the wireless multimedia applications was presented. In this scheme, an optimization problem is formulated with fairness constraints for maximizing system capacity and resource utilization. The solution of this problem yields the optimal allocation of sub-channel, path and power. A heuristic algorithm to solve the optimization problem was also presented to ensure that the adaptive resource allocation is performed efficiently in runtime. In [96], a resource allocation approach for embedded multimedia systems using heterogeneous multiprocessors was presented, in which optimal resources are allocated to each application to meet its throughput requirement. Synchronous Dataflow Graphs (SDFG) are used to model multimedia applications with time and resource constraints and to find the optimal resource allocation.

QoS estimation according to the system activities and resource status has been studied in many ways. A QoS model of a router with feedback control that monitors the state of resource usage and adaptively adjusts parameters of traffic admission control to estimate QoS and resource utilization was presented in [97]. Batch Scheduled Admission Control (BSAC) method to predict service delay for high priority jobs in Internet-type networks was presented in [98]. A regression-based model for dynamically provisioning resource demand to deliver given QoS expectation was presented in [99]. An adaptive model for the tradeoff between service performance and security in service-based environments was presented in

[100]. This model can be used to adjust security configurations to provide sufficient protection and satisfy service performance requirements with limited system resources.

For adaptive resource allocation for QoS management in SBS, application level differentiated services [101, 102] were introduced to control QoS for different classes of service consumers. When the system resources are limited, fewer resources are allocated for normal consumers, and most resources are reserved for satisfying premium consumers' expected QoS. Feedback controlled web services [103, 104] were developed to adjust QoS to meet the most important performance when resources are limited and consumers' required performance cannot be fully satisfied. In [105], an integrated QoS management approach in SBS in order to satisfy users' QoS requirements by providing differentiated system resources and priority of workflows was presented. In [106, 107], a general methodology for developing SBS with QoS monitoring and adaptation was introduced. This methodology supports monitoring of the changing system status, analysis and control of tradeoffs among multiple QoS features, and adapting its service configuration to satisfy multiple QoS requirements simultaneously.

Existing resource allocation approaches cannot support dynamically changing runtime environments in SBS, such as workflow composition, QoS requirements, workflow priorities and resource status.

7.2 Overview of an Approach to Adaptive Resource Allocation for SBS

This section presents a resource allocation approach to adaptively allocating the system resources of servers to their services in runtime in order to satisfy the throughput requirements of the multiple workflows in SBS.

In SOA, a service that cannot be decomposed to smaller services and serves only one type of service-request is considered as an *atomic service*. All other services are considered as *composite services*. In order to have adaptive resource allocation capabilities in SBS, the SBS need to analyze the relationship between resource allocation of each server to its atomic services and the throughputs in both atomic and composite services provided by the server. In my approach, I first developed the Resource-Allocation-Throughput (RAT) model for an atomic service, and extend the model for the entire SBS to analyze the relationship between resource allocation and throughputs of the multiple workflows in SBS. The details of RAT model will be presented in Section 7.3. Based on the RAT model, I will present an algorithm to automatically formulate a linear programming optimization problem [108] to find the optimal resource allocation to serve the users' service requests in Section 7.5. The definition of the optimal resource allocation is as follows:

1) Optimal resource allocation will serve all the service requests of workflows in SBS until system resources are exhausted.
2) If system resources are exhausted and all the service requests cannot be served, the optimal resource allocation will selectively serve the service requests to

maximize the throughput of SBS subject to the minimum throughput requirements and priorities of workflows.

The major distinction between resource allocation approach and other resource allocation approaches discussed before is that my approach can maximize utilization of limited system resources to reach maximum throughput according to dynamic workflow composition, throughput requirements, workflow priorities and resource status, rather than just serving users' service-requests in first-in-first-serve manner by matching available system resources upon the arrival of the user requests and requirements.

The conceptual view of resource allocation approach is depicted in Figure 10. Each rectangle represents a discrete system component. Functional details and relationships among the components are described as follows:

- **Workflow Manage**r dynamically creates workflows based on the users' requests and available services. Automatic service discovery and composition can be done with semantic service description [109].
- **Workflow Monitor** is responsible for gathering information about users' service-request rates of each workflow from the Workflow Manager. The users' service-request rates can be dynamically changed in runtime, and the monitoring of service-request rates is done periodically.
- **Resource Consumption Estimator** is responsible for estimating the amount of resource to be consumed for executing service-requests of each workflow. The estimation is done based on the RAT model, which will be discussed in Section 7.3.

- **Resource Monitor** is responsible for collecting information about available resources of each server in SBS. The resource monitoring module for a large SBS can be built at the middleware layer, such as Ganglia [110] and Resource Broker [111, 112].
- **Resource Manager** is responsible for finding an optimal resource allocation that maximizes the throughput of SBS with the constraints on the throughput requirements, available system resources, workflow orders, priorities, and service-request rates. In order to find the optimal resource allocation, I use linear programming (LP) for optimization of a linear objective function, subject to linear inequality constraints [113]. Detailed discussion on formulating linear programming optimization problem will be given in Section 7.5. Resource Manager is also responsible for adaptively allocating system resources of servers. The resource allocation can be dynamically done in runtime at the middleware layer, such as QoS Resource Manager [114] and Globus [115].

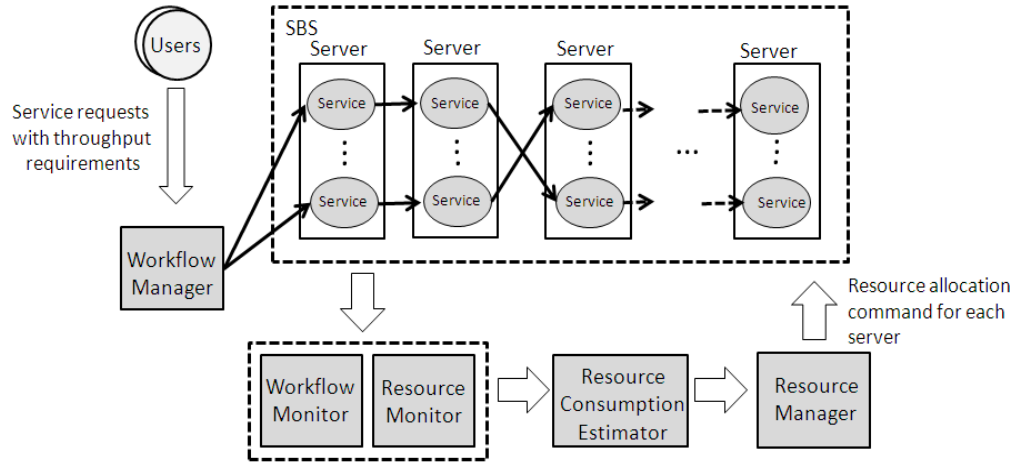

Figure 10: Conceptual View of Dynamic Adaptive Resource Allocation for SBS

The resource allocation to satisfy users' throughput requirements can be determined as follows:

An Approach to Dynamic Adaptive Resource Allocation in SBS

Step1) (Initialization) Service providers, users, workflow manager, and resource monitors send the following information to the resource manager

- Service Providers: the RAT model of each service in SBS.
- Users: the throughput requirements for the workflows
- Workflow Manager: the structure of each workflow, priority of each workflow, service request rate of each workflow
- Each Resource Monitors of each server: the amount of available server resource

Step2) (in run-time) The resource manager formulates a linear programming (LP) problem subject to the following constrains and objective:

- Constraint 1) The throughput of a workflow cannot exceed the service-request rate of the workflow
- Constraint 2) The throughput of a workflow must satisfy the minimum throughput requirements specified by the user
- Constraint 3) The total amount of resources consumed by all services in a server cannot exceed the amount of resources of the server
- Objective) Allocating server resources to support services for maximizing the total throughput of all workflows in SBS

The algorithms for formulating and solving the LP problem will be presented in Section 7.5.

Step3) Resource allocation is determined by solving the LP problem generated in *Step2)*. Resource manager sends resource allocation command to each server in SBS.

In this approach, the five challenges discussed in Section 1.2.2 are addressed as follows: the capability of situation awareness is provided by workflow manager, workflow monitor and resource monitor. Context analysis and QoS prediction are provided by resource consumption estimator. Resource adaptation in run-time is provided by resource manager. Efficiency and scalability of algorithms for formulating and solving a LP problem will be discussed in Section 7.6.

7.3 RAT models For Atomic Services in SBS

In this section, I will discuss RAT model for the relationship between system resource allocation of servers and the throughput of SBS. Before developing the RAT model of the entire SBS, it is needed to model the relationship between resource allocation of each server to its atomic services and the throughput of the atomic services.

RAT model for an atomic service is shown in Figure 11. Limited system resources of a server are allocated to an atomic service. As service-requests arrive at the Service Request Queue, the atomic service creates multiple threads, which utilize the system resource to process the service-requests and send out the service responses. In this model, I consider the following five factors to estimate the amount of resource to be consumed by the atomic service for executing service-requests:

- **Service-request rate R of an atomic service**: R is the average number of service-requests per second arriving at an atomic service, and represents the workload of the atomic service. An atomic service can limit the number of process threads and adaptively drop some service requests or put them into a waiting-queue according to the resource status of the atomic service in order to maintain a certain level of throughput.
- **Critical resource**: Processing service-request will require various types of system resources of a server, such as CPU time, memory and network bandwidth. As R increases and more threads are created, one of the system resources of the server will become a bottleneck, and such system resource of the server is called the critical resource of the server.
- **Percentage of Allocated critical resource A**: A is the percentage of allocated critical resource of a server to an atomic service provided by the server over the total available critical resource of the server. I only need to consider the allocation of the critical resource of a server because the critical resource of the server becomes a bottleneck first when other resources are sufficient.
- **Throughput P of an atomic service**: P is defined as the average number of service responses per second of the atomic service. P is the sum of the service-responses of each thread per second in an atomic service, and determined by the R and A of the atomic service.
- **Throughput requirement of a workflow**: This is the minimum number of service responses per second required by the users for a workflow.

From this model, notice that

1) P = R if A is not exhausted. In this case, P will be determined by R.

2) Increasing R will increase P until allocated resource is exhausted.

3) When A is exhausted, P will not increase even if R increases. Some of service-requests will be dropped or put into a waiting-queue. In this case, P will be determined by A.

Based on the above observations, estimate the throughput P of an atomic service as follows:

$$P = R \quad \text{when A is not exhausted}$$

$$= \alpha A \quad \text{when A is exhausted}$$

where α is a proportional constant, which is different for different atomic services. Given the allocated critical resource A, αA is the maximum throughput of the atomic service. Define the service-cost of an atomic service as follows:

Definition: The *service cost* of an atomic service is the percentage of critical resource required to serve a service-request over the total available critical resource.

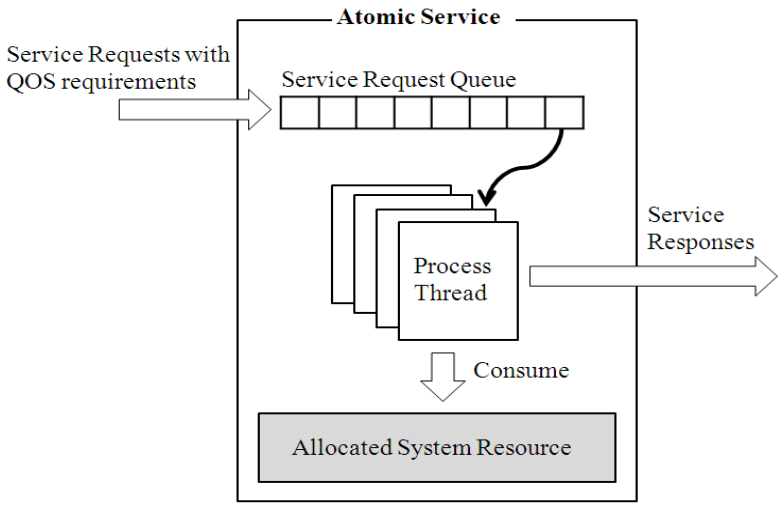

Figure 11: The RAT Model for an Atomic Service

It is noted that since $P = \alpha A$ when A is exhausted, the service cost of an atomic service is $A / \alpha A = 1/\alpha$.

The critical resource type and the service cost of an atomic service can be estimated from the performance models for QoS related cause-effect dynamics in SBS, called *Activity-State-Event-QoS (ASEQ)* models [116, 117]. It is necessary for developing a general methodology for constructing the ASEQ models for SBS by conducting well-designed experiments in SBS and collecting data related to various users' service-requests and system resource states. In this dissertation, I only need to consider the relationship between throughput and resource allocation since this research only considers the throughput. The service cost of an atomic service can be estimated as follows:

S1) Allocate an arbitrary amount of critical resource A to a target atomic service.

S2) Increase the R and measure the P of the atomic service until the critical resource of the server is exhausted.

S3) Repeat S1) and S2) to obtain P with various A and R.

S4) Using the statistical linear regression analysis on the P obtained in S3), estimate the service cost of the target atomic service

7.4 RAT models For Composite Services in SBS

SBS consists of multiple servers and workflows. Each server provides multiple atomic services, and each atomic service is represented by its service cost. Each workflow is composed of atomic services in different servers. Each

server has its critical resource and ability to adaptively allocate the critical resource to its atomic services in runtime.

<SBS> ::= “(” {<Workflow>}$^{+}$, {<Server>}$^{+}$ “)”
<Server> ::= “(” <Server ID>, <Critical resource>, < % of available critical resource>,
{<Atomic service>}$^{+}$ “)”
<Atomic service> ::= “(” <Service ID>, <Service cost> <Server ID> “)”
<Workflow> ::= “(“ {<Service Composition>}$^{+}$, <Service-request rate>,
<Throughput-requirement> <Priority> “)”
<Starting> ::= <Atomic service>
<Ending> ::= <Atomic service>
<Service Composition> ::= <Sequence> | <Parallel Sprit> | <Merge> |
<Pick>
<Sequence> ::= “(” <Starting>, <Ending> “)”
<Parallel Sprit> ::= “(” <Starting>, {<Ending>}$^{+}$ “)”
<Merge> ::= “(” {<Starting>}$^{+}$, <Ending> “)”
<Pick> ::= “(” <Starting>, <Condition>, {<Ending>}$^{+}$ “)”

In practice, it may be impossible to satisfy all the users’ throughput requirements simultaneously because of limited system resources of servers. In this case, the system needs to relax the throughput requirements for some workflows. This can be done by setting priorities for workflows and allocate more resources to high-priority workflows when system resources are exhausted.

Based on the relations among servers, atomic services and workflows, describe a SBS in BNF. From the RAT model of atomic service and the BNF of

SBS, calculate the throughput of each workflow with given service-cost, percentage of allocated critical resource and service-request rate for each workflow as follows:

Calculating throughput of a workflow in SBS

---

1. S = {s | s is an atomic service in a workflow}
2. For each s ∈ S
3. α = service-cost of s
4. R = service-request rate of s
5. A = Percentage of allocated critical resource to s
6. Let P(s) be the throughput of s
7. if ($R \leq A / \alpha$) then P(s) = R
8. else P(s) = A / α
9. Throughput of the workflow = MINIMUM s ∈ S (P(s))

7.5 Resource allocation using RAT models and linear programming

Based on the BNF of SBS and the RAT model of atomic service, formulate a constraint-optimization problem using linear programming. The solution of this optimization problem determines the amount of critical resource required for each atomic service. The following pseudo code shows algorithm to automatically formulate a linear programming optimization problem with given BNF of a SBS.

Formulating a linear programming problem to allocate critical resources of all servers in a SBS

1. W = {w | w is a workflow in SBS}
2. Sv = {sv | sv is a server in SBS}
3. For each w ∈ W
4. Let throughput of w be TH(w)
5. Pr(w) = Priority of w
5. Add objective function of LP ("Maximize $\sum_{w \in W}(TH(w) \times Pr(w))$")
6. TR(w) = throughput-requirement of w
7. SR(w) = service-request rate of w
8. Add constraint ("TH(w) ≤ Service-request rate of w")
9. If (TR(w) ≤ SR(w))
10. Add constraint ("TH(w) ≥ TR(w)")
11. For each sv ∈ Sv
12. For each w ∈ W
13. If w uses atomic services provided by sv
14. C = 0
15. For each atomic service s in sv used by w
16. α = service-cost of s
17. C = C + α×TH(w)
18. AR(sv) = % of available critical resource of sv
19. Add constraint ("C ≤ AR(sv)")

7.6 Adaptive Allocation of Critical Resources of Servers for SBS

Using the Simplex algorithm [118], solve the linear programming problem presented in Section 7.5 and find the optimal throughput of each workflow in SBS. Given n workflows and m servers, this problem will have m variables and 2m+n constraints. Thus, the complexity of solving this problem is O(m3 + nm2) in the

worst case. On the average, these linear programming problems are solved in polynomial time [119]. Hence my approach addresses the challenge of efficiency and scalability discussed in Section 1.2.2.

Once the throughput of each workflow is found, allocate the critical resource to each atomic service so that each workflow can produce its optimal throughput. The following pseudo code shows how to allocate the critical resources of servers to their atomic services.

An algorithm for allocating critical resources of servers to their atomic services

---

W = {w | w is a workflow in SBS}
Sv = {sv | sv is a server in SBS}
S = { s | s is an atomic service in Sv}
Solve the LP problem using the Simplex algorithm and find optimal throughputs of workflows
OT(w) = optimal throughput of workflow w ∈ W
For each s ∈ S
  α = service-cost of s
  Let A(s) be the amount of critical resource to be allocated to s
  A(s) = 0
  For each workflow w that uses s
    A(s) = A(s) + OT(w) × α

7.7 A Demonstration of Adaptive Resource Allocation in SBS

This section demonstrates resource allocation approach using a SBS shown in Figure 12 with four servers, nine atomic services and three workflows. The four servers were implemented as virtualized platforms using Hyper-V Manager 6.0. The configurations of the servers are as show in Table 5.

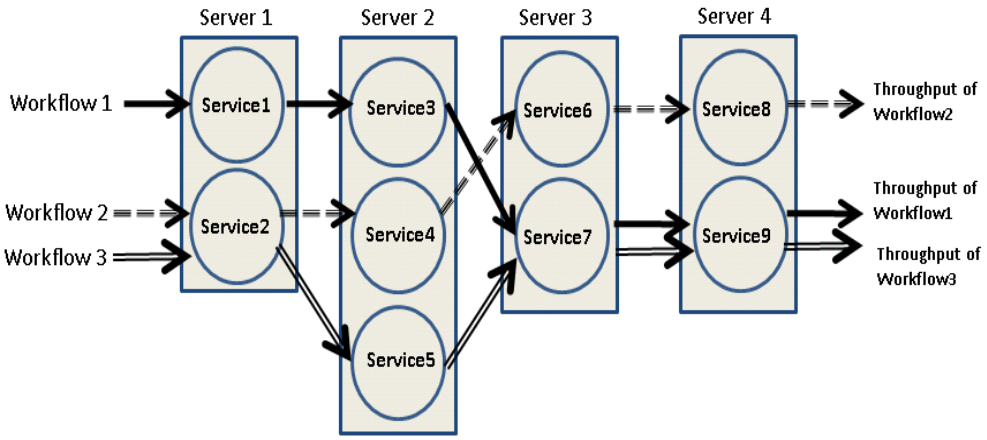

Figure 12: A SBS Demonstration System

Table 5: Configurations of the Four Servers in Demonstration System

Server	Operating System	CPU	Memory
Server 1	Windows Server 2003	0.7 GHZ	1024 MB
Server 2	Windows Server 2003	2.94 GHZ	256 MB
Server 3	Windows Server 2003	0.7 GHZ	1024 MB
Server 4	Windows Server 2003	0.7 GHZ	1024 MB

The nine atomic services were implemented as web services using C# Microsoft .NET framework. Using the RAT model of atomic service discussed in Section 7.3, the nine atomic services create threads to process runtime service requests from multiple users simultaneously. The atomic services are listed as follows:

- Service 1 (Video Encoding Service): It takes 35 KB binary stream as input and converts it to AVI video format.
- Service 2 (Data Compression Service): It takes 35 KB byte array as input and compresses the data using Data Compressing library provided by C#.

- Service 3 (DES Encryption Service): It takes any size of byte array as input and encrypts the data with key size 64 bits using DES Encryption library provided by C#.
- Service 4 (AES128 Encryption Service): It takes any size of byte array as input and encrypts the data with key size 128 bits using AES Encryption library provided by C#.
- Service 5 (AES256 Encryption Service): It takes any size of byte array as input and encrypts the data with key size 256 bits using AES Encryption library provided by C#.
- Service 6 (DSA Digital Signature Service): It takes any size of byte array as input and produces digital signature with key size 1024 bits using DSA Digital Signature library provided by C#
- Service 7 (MD5 Hashing Service): It takes any size of byte array as input and generates hash of the data using MD5 Hashing library provided by C#.
- Service 8 (DSA Digital Signature Service): It takes any size of byte array as input and produces digital signature with key size 2048 bits using DSA Digital Signature library provided by C#
- Service 9 (SHA1 Hashing Service): It takes any size of byte array as input and generates hash of the data using SHA1 Hashing library provided by C#.

Besides these services, I also developed a throughput monitoring service with Windows Performance Objects which collects the throughput of each workflow and the resource consumption of each server in runtime.

To estimate the service cost of each atomic services, I ran one set of experiments. In each set of the experiments, I invoked its atomic service and increased service request rate until the system resource of the server was exhausted. The throughput of each atomic service using the throughput monitoring service is shown in Figure 13. The shapes of these curves are similar because the service response rates of the atomic services did not increase after the system resource of the server was exhausted even if the service request rates increased.

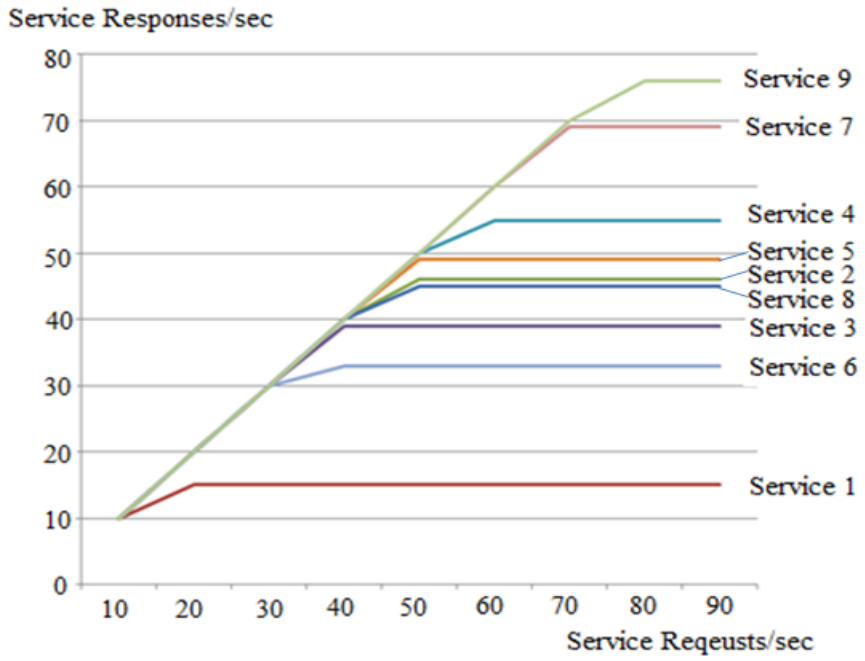

Figure 13: The Throughput Threshold of Each of the Nine Atomic Services in Demonstration System

The critical resource of the server of each atomic service is observed using the throughput monitoring service and shown in Table 6. Based on the experimental

results and the RAT model, I found the service cost of each atomic service using the process described in Section 7.3, which is shown in Table 7.

Table 6: The Critical Resource of Each Server in Demonstration System

Server	Server 1	Server 2	Server 3	Server 4
Critical Resource	CPU	Memory	CPU	CPU

Table 7: The Service Cost of Each Atomic Service of Demonstration System

Service	1	2	3	4	5	6	7	8	9
Service Cost	6.7%	2.1%	2.6%	2%	2.1%	2.94%	1.4%	2.17%	1.3%

There are three workflows composed by the nine atomic services as shown in Figure 12. I ran two experiments on these workflows. In the first experiment, the workflows were executed without using resource allocation approach. In the second experiment, I ran the workflows with resource allocation approach. The service request rate of each workflow was set to 20 service requests/sec. The throughput requirements of workflow 1, workflow 2 and workflow 3 were set to 3, 15 and 10 service responses/sec respectively. Based on the critical resource of each server from the throughput monitoring service and the estimated service cost of each atomic service, the BNF representation of the SBS is given below:

<service1> = (<“service1”> <6.7%> <“server1”>)

<service2> = (<“service2”> <2.1%> <“server1”>)

<service3> = (<“service3”> <2.1%> <“server2”>)

<service4> = (<“service4”> <2%> <“server2”>)

<service5> = (<“service5”> <2.1%> <“server2”>)

<service6> = (<“service6”> <2.94%> <“server3”>)

<service7> = (<“service7”> <1.4%> <“server3”>)

<service8> = (<“service8”> <2.17%> <“server4”>)

<service9> = (<“service9”> <1.3%> <“server4”>)

<Server1> = (<“server1”> <CPU> <100%> <service1> <service2>)

<Server2> =

(<“server2”><Memory><100%><service3><service4><service5>)

<Server3> = (<“server3”> <CPU> <100%> <service6> <service7>)

<Server4> = (<“server3”> <CPU> <100%> <service8> <service9>)

<Sequence1> = (<service1><service3>)

<Sequence2>=(<service3><service7>)

<Sequence3> = (<service2><service4>)

<Sequence4>=(<service4><service6>)

<Sequence5> = (<service2><service5>)

<Sequence6>=(<service5><service7>)

<Sequence7>=(<service6><service8>) <Sequence8>=(<service7><service9>)

<Workflow1>=((<Sequence1><Sequence2><Sequence8>)<30 /sec> <3 /sec>

<1>)

<Workflow2>=((<Sequence3><Sequence4><Sequence7>)<30 /sec><15 /sec>

<1>)

<Workflow3>=((<Sequence5><Sequence6><Seqiemce8>)<30 /sec><10 /sec>

<1>)

From the BNF representation of the SBS, the linear programming problem using the algorithm described in Section 7.5 is generated, shown as follows:

Maximize p = TH1 + TH2 + TH3 Subject to

TH1 <= 15, TH2 <= 15, TH3 <= 15, TH1 >= 3, TH2 >= 15, TH3 >= 10

6.7TH1 + 2.1TH2 + 2.1TH3 <= 100

2.6TH1 + 2 TH2 + 2.1 TH3 <= 100

1.4TH1 + 2.94TH2 + 1.4TH3 <= 100

2.17TH1 + 1.3TH2 + 1.3TH3 <= 100

After solving the linear programming problem, the throughputs for workflow 1, workflow 2 and workflow 3 were 3, 26.6 and 12.4 responses/sec, respectively. Then, I estimated the optimal resource allocation of the critical resource of each server using the resource allocation algorithm presented in Chapter 8, which is given in Table 8. The throughputs of the three workflows of the two experiments are shown in Figure 14. It is noted that there were significant differences in throughputs of the workflows between the two experiments. It is also noted that the throughput requirements of workflow 2 and workflow 3 were not satisfied in the first experiment which was run without using resource allocation approach, but all the throughput requirements were satisfied in the second experiment using resource allocation approach, and that the overall throughput of the SBS, the sum of the throughputs of these three workflows increased by 28%.

Table 8: Resource Allocation Results in the Demonstration System

	CPU time of Server 1		Memory Access time of Server 2			CPU time of Server 3		CPU time of Server 4	
Service	1	2	3	4	5	6	7	8	9
Optimal Resource Allocation	21%	79%	15%	55%	27%	30%	70%	20%	80%

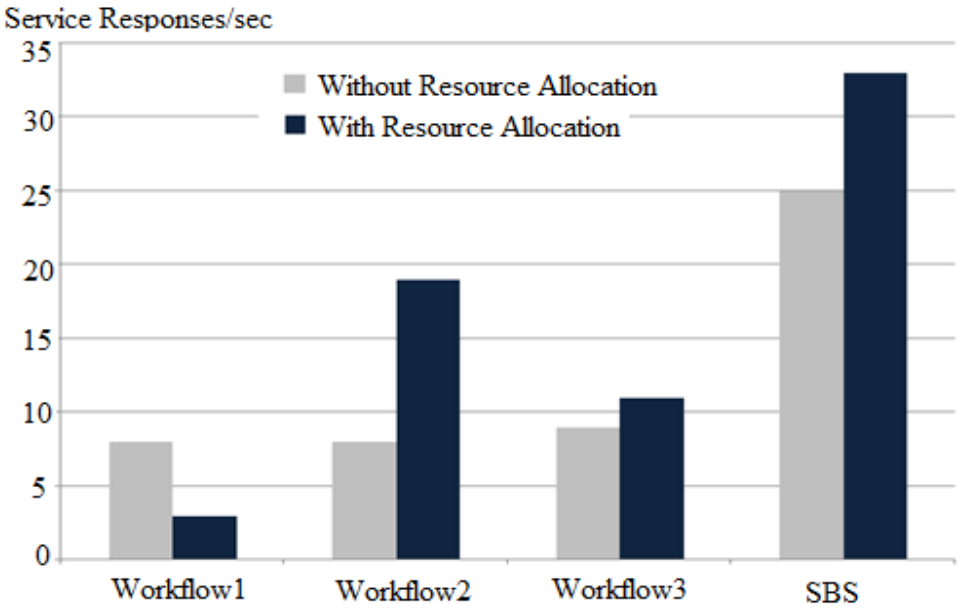

Figure 14: The Throughputs of Workflows in Demonstration System

CHAPTER 8

CONCLUSION AND FUTURE WORKS

8.1 Summary of Research Contributions

In this dissertation, two interrelated problems of service-based systems (SBS) are addressed: protecting users' data confidentiality from service providers, and managing performance of multiple workflows in SBS.

To address the first problem, I have presented an approach to protecting the confidentiality of users' data from unauthorized entities, including service providers of SBS. In this approach, users' data confidentiality is protected by preventing the service providers from satisfying the following two conditions simultaneously: *Condition 1)* The service providers know where the users' confidential data is located in SBS and have the privilege to access and collect the users' confidential data in SBS, and *Condition 2)* The service providers understand the meaning of the users' data. Data obfuscation techniques, anonymous service usage protocol (ASUP), and techniques for software module decomposition and distributed execution in SBS are developed in order to prevent the service providers from satisfying the two conditions simultaneously.

To address the second problem, I have presented an approach to adaptively allocating system resources of servers in a SBS to their services in runtime in order to satisfy the performance requirements of multiple workflows in the SBS. This approach analyzes the relationship between resource allocations and performance of workflows in the SBS and then uses linear programming to determine resource allocation to satisfy users' requirements on performance of

each workflow. My dynamic resource allocation approach has been demonstrated and shown to result in substantially increase the throughput of SBSs.

8.2 Future Research

In the approach to protecting the confidentiality of users' data, the software service providers and infrastructure providers in SBSs are separated into different domains. From the business point of view, there are some complications or limitations on separating software providers and infrastructure providers required in my SBS architecture because most of current SBS service providers provide both software and infrastructure. It is desirable to investigate the economical impacts as well as possible changes of industry practices for adopting this SBS architecture.

In my approach to software module decomposition and distributed execution, a requirement specification language to specify users' data confidentiality requirements is developed. Currently, the specification language can be used only for specifying static data confidentiality requirements, in which the sensitivity of users' data does not change in run-time. However, the sensitivity of users' data may be dynamically changed in run-time due to changes in execution environments. Further research is needed to extend my requirement specification language to support specification of dynamic confidentiality requirements in run-time.

The anonymous service usage protocol (ASUP) developed for hiding users' real identities in SBS only supports simple access control policies, such as

discretionary access control (DAC) policies. Since the simple access control policies are solely based on the authentication of user's identity, my ASUP cannot support systems requiring more powerful access control policies, such as mandatory access control (MAC) policies and role-based access control (RBAC) policies. Further research is needed to extend ASUP to support more flexible and comprehensive access policies.

Currently, my resource allocation approach is to improve the use of available resource to satisfy the performance of workflows. However, other QoS features, such as service delay, accuracy and jitter, may also important for the workflows of certain applications. Further research is needed to extend my resource allocation approach to supporting multiple QoS features.

REFERENCES

[1] G. Alonso, F. Casati, H. Kuno, and V. Machiraju, "Web Services - Concepts, Architectures and Applications", M. J. Carey and S. Ceri (eds.), Springer, 2004.

[2] S. S. Yau, H. Davulcu, S. Mukhopadhyay, D. Huang, Y. Yao and H. Gong, "Adaptable Situation-Aware Secure Service-Based (AS3) Systems", *Information Security Research*, C. Wang and S. King (eds.), 2007, Chapter 5, pp. 585-596.

[3] S. S. Yau, M. Yan, and D. Huang, "Design of Service-Based Systems with Adaptive Tradeoff Between Security and Service Delay", *Proceedings of 4th International Conference on Autonomic and Trusted Computing*, 2007. pp. 103-113.

[4] J. Heiser and M. Nicolett, "Assessing the security risks of cloud computing," Gartner Report, 2009, http://www.gartner.com/DisplayDocument?id=685308

[5] G. Brahnmath, R. R. Raje, A. Olson, M. Auguston, B. R. Bryant, and C. C. Burt, "A Quality of Service Catalogue for Software Components", *Proceedings of Conference on Southeastern Software Engineering*, 2002, pp. 513–520.

[6] J. Ivan, H. Caroline and F. Stephane, "A Comprehensive Quality Model for Service-Oriented Systems", *Software Quality Control*, Vol. 17(1), 2009, pp. 65-98.

[7] D. Menasce, "QoS issues in Web services", *IEEE Internet Computing*, Vol. 6(6), 2002, pp. 72-75.

[8] J. B. Horrigan, "Use of cloud computing applications and services," *Pew Internet & American Life project memo*, 2008.

[9] A. Erradi and P. Maheshwari, "A Broker-Based Approach for Improving Web Services Reliability", *IEEE International Conference on Web Services (ICWS'05)*, 2005, pp. 355-362.

[10] S. S. Yau, J. Huang and Y. Yin, "Improving the Trustworthiness of Service QoS Information in Service-based Systems", *Proceedings of 7th Conference on Autonomic and Trusted Computing (ATC)*, 2010, pp. 208-218.

[11] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu and S. S. Yau, "Efficient Provable Data Possession for Hybrid Clouds", *Proceedings of 17th ACM Conference on Computer and Communications Security*, 2010, pp.756-758.

[12] Y. Zhu, H. Wang, Z. Hu, G.-J. Ahn, H. Hu and S. S. Yau, "Dynamic Audit Services for Integrity Verification of Outsourced Storages in Clouds",

Proceedings of 26th ACM Symposium on Applied Computing, 2011, pp. 34-41.

[13] A. Celesti, "Security and Cloud Computing: InterCloud Identity Management Infrastructure", *Proceedings of 19th International Workshop on Enabling Technologies: Infrastructures for Collaborative Enterprises*, 2010, pp. 263-265.

[14] H. Chang and C. Euiin, "User Authentication in Cloud Computing", *Communications in Computer and Information Science*, Vol. 151(1), pp. 338-342.

[15] Y. Zhu, G-J. Ahn, H. Hu, S. S. Yau, H. G. An and S. Chen, "Dynamic Audit Services for Outsourced Storages in Clouds", *IEEE Transactions on Services Computing*, Vol. PP (99), August, 2011, pp. 1-13.

[16] S. S. Yau and J. Huang, "A User-Centric Approach to Assessing Confidentiality and Integrity of Service-Based Workflows", *Proceedings of 3rd International Conference on Human-centric Computing (HumanCom)*, 2010, pp.89-94.

[17] C. Yang, C. Lin and S. Chen, "A Workflow-based Computational Resource Broker with Information Monitoring in Grids", *Proceedings of 5th International Conference on Grid and Cooperative Computing*, 2006, pp. 105-206.

[18] C. T. Yang, P. C. Shih, C. F. Lin and S. Y. Chen, "A Resource Broker with an Efficient Network Information Model on Grid Environments", *The Journal of Supercomputing*, Vol. 40(3), 2007, pp. 249-267.

[19] S. Venugopal, X. Chu and R. Buyya, "A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol", *Proceedings of 16th International Workshop on Quality of Service (IWQoS 2008)*, 2008, pp. 78-89.

[20] C. C. Aggarwal, "On k-anonymity and the curse of dimensionality," *Proceedings of Very Large Data Base*, 2005, pp. 901-909.

[21] K. LeFevre, D. 1.DeWitt, and R. Ramakrishnan, "Incognito: Efficient full-domain k-anonymity," *Proceedings of the ACM SIGMOD International Conference on Management of Data*, 2005, pp. 49-60.

[22] Mowbray M, Pearson S, "A client-based privacy manager for cloud computing", *Proceedinging of Conference on Communication System Software and Middleware*, 2009. 138-145.

[23] R. Bayardo and R. Agrawal, "Data privacy through optimal kanonymization", *Proceedings of International Conference on Data Engineering*, 2005, pp. 217-228.

[24]Format-Preserving Encryption, Voltage Security, 2009, http://www.voltage.com/technology/Technology_FormatPreservingEncryption.htm

[25] Case Studies: TC3 Health, Amazon Web Services LLC, 2009, http://aws.amazon.com/solutions/casestudies/tc3-health/

[26] C. Yang, C. Lin and S. Chen, "A Workflow-based Computational Resource Broker with Information Monitoring in Grids", *Proceedings of 5th International Conferences on Grid and Cooperative Computing*, 2006, pp. 105-206.

[27] S. Venugopal, X. Chu and R. Buyya, "A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol", *Proceedings of 16th International Workshop on Quality of Service (IWQoS 2008)*, 2008, pp. 34-41.

[28] C. T. Yang, P. C. Shih, C. F. Lin and S. Y. Chen, "A Resource Broker with an Efficient Network Information Model on Grid Environments", The Journal of Supercomputing, vol. 40(3), 2007, pp. 249-267.

[29] R. Buyya, C. S. Yeo, and S. Venugopal, "Market Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", *Proceedings of 10th IEEE International Conference on High Performance Computing and Communications*, 2008, pp. 234-242.

[30] P. Lardieri, J. Balashbramanian and D. C. Schmidt, "A multi-layered resource management framework for dynamic resource management in enterprise DRE systems", *Journal of Systems and Software*, Vol. 80(7), 2007, pp. 984-996.

[31] P. E. Hladik, H. Cambazard, A. M. Deplanche and N. Jussien, "Solving a real-time allocation problem with constraint programming", *Journal of Systems and Software*, Vol. 81(1), 2008, pp. 132-149.

[32] J. N. Hooker and G. Ottoson, "Logic-based benders decomposition", *Mathematical Programming*, Vol. 96, 2003, pp. 33-60.

[33] J. Stoesser, C. Roessle and D. Neumann, "Decentralized Online Resource Allocation for Dynamic Web Service Applications", *Proceedings of 4th International Conferences on Enterprise Computing, E-Commerce, and E-Service*, 2007, pp. 425-428.

[34] M. Mazzucco, I. Mitrani, J. Palmer, M. Fisher and P. McKee, "Web Service Hosting and Revenue Maximization", *Proceedings of 5th European Conferences on Web Services (ECOWS'07)*, 2007, pp. 92-98.

[35] M. Kallits, G. Michailidis and M. Devetsikiotis, "Pricing and Optimal Resource Allocation in Next Generation Network Services", *Proceedings of IEEE Sarnoff Symposium*, 2007, pp. 1-5.

[36] F. Meshkati, H. V. Poor and S. C. Schwartz, "Energy-Efficient Resource Allocation in Wireless Networks", *Signal Processing Magazine*, Vol. 24(3), 2007, pp. 58-68.

[37] C. Bae and D. Cho, "Fairness-Aware Adaptive Resource Allocation Scheme in Multihop OFDMA Systems", *IEEE Communications Letters*, Vol. 11(2), 2007, pp. 134-136.

[38] S. Stuijk, T. Basten, M. Geilen and H. Corporaal, "Multiprocessor Resource Allocation for Throughput Constrained Synchronous Dataflow Graphs", *Proceedings of Conferences on the 44th IEEE annual Design Automation*, 2007, pp. 777-782.

[39] Z. Yang, N. Ye and Y.-C. Lai, "QoS model of a router with feedback control", *Quality and Reliability Engineering*, Vol. 22(4), 2006, pp. 429-444.

[40] N. Ye, B. Harish, X. Li and T. Farley, "Batch scheduled admission control for service dependability of computer and network resources", *Information, Knowledge, Systems Management*, Vol. 5(4), 2006, pp. 211-226.

[41] Q. Zhang, L. Cherkasova and E. Smirni, "A Regression Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications", *Proceedings of the 4th International Conference on Autonomic Computing*, 2007, pp. 27-36.

[42] S. S. Yau, Y. Yin and H. G. An, "An Adaptive Model for Tradeoff between Service Performance and Security in Service-based Environments", *Proceedings of International Conference on Web Services (ICWS 2009)*, 2009, pp. 287-294

[43] L. Eggert and J. Heidemann, "Application-Level Differentiated Services for Web Services", *Journal of World-Wide Web*, Vol. 2(3), 1999, pp. 133-142.

[44]. G. Rao and B. Ramamurthy, "DiffServer: Appliction Level Differentiated Services for Webservers," *Proceedings of IEEE International Conference on Communication*, Vol. 5, 2001, pp. 1633-1637.

[45] C. Lu, Y. Lu, T.F. Abdelzaher, J.A. Stankovic, and S.H. Son, "Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers," *IEEE Transactions on Parallel and Distributed Systems*, Vol. 17(9), 2006, pp 1014-1027.

[46] T.F. Abdelzaher, J.A. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback Performance Control in Software Services," *IEEE Control Systems Magazine*, Vol. 23(3), 2003, pp. 74–90.

[47] G. Wang, A. Chen, C. Wang, C. Fung and S. Uczekaj, "Integrated quality of service (QoS) management in service-oriented enterprise architectures", *Proceedings of the 8th International Conference on Enterprise Distributed Object Computing*, 2004, pp. 21-32.

[48] S. S. Yau, N. Ye, H. Sarjoughian and D. Huang, "Developing Service-based Software Systems with QoS Monitoring and Adaptation", *Proceedings of the 12th International Workshop on Future Trends of Distributed Computing Systems*, 2008, pp. 74-80.

[49] S. S. Yau, N. Ye, H. Sarjoughian, D. Huang, A. Roontiva, M. Baydogan and M. Muqsith, "Towards Development of Adaptive Service-based Software Systems", *IEEE Transactions on Service Computing*, 2009.

[50] DoD Trusted Computer System Evaluation Criteria, http://csrc.nist.gov/publications/history/dod85.pdf

[51] S. S. Yau and H. G. An, "Confidentiality Protection in SBS Computing Systems," *International Journal of Software Informatics*, Vol. 4(4), 2010, pp. 351-365.

[52] M. Iwaihara, K. Murakami, G.-J. Ahn and M. Yoshikawa, "Risk Evaluation for Personal Identity Management Based on Privacy Attribute Ontology," *Proceedings of 27th International Conference on Conceptual Modeling (ER 2008)*, 2008, pp 183-198.

[53] M. Mateas and N. Michael, "A Box, Darkly: Obfuscation, Weird Languages, and Code Aesthetics", *Proceedings of 6th Digital Arts and Culture Conference*, 2005, pp. 144-153.

[54] L. Ertaul and S. Venkatesh, "Novel obfuscation algorithms for software security", *Proceedings of International Conference on Software Engineering Research and Practice*, 2005, pp. 209–215.

[55] W. Dong and H. Yu, "Web Service Testing Method Based on Fault-coverage", *Proceedings of 10th IEEE on International Enterprise Distributed Object Computing Conference Workshops*, 2006, pp. 43-49.

[56] J. Gibson, "Developing A Requirements Specification For A Web Service Application", *Proceedings of 12th IEEE International Conference on Requirements Engineering*, 2004, pp. 340-344.

[57] Kona S, Bansal A, Gupta G, Hite T, "Web Service Discovery and Composition using USDL", *Proceedings of 3rd IEEE International Conference on E-Commerce Technology*, 2006, 65–67.

[58] A. Damodaram and H. Jayasri, "Authenticatio without Identification using Anonymous Credential System", *International Journal of Computer Science and Information Security (IJCSIS*), Vol. 3(1), 2009, pp. 34-37.

[59] S. S. Yau, L. Zhu, D. Huang and H. Gong, "An Approach to Automated Agent Deployment in Service-based Systems", *Proceedings of 10th IEEE International Symposium on Object and Component-Oriented Real-Time Distributed Computing (ISORC)*, 2007, pp. 257-264.

[60] D. Bakken, R. Parameswaran, D. Blough, A. Franz, Ty J. Palmer, "Data Obfuscation: Anonymity and Desensitization of Usable Data Sets," *IEEE Security and Privacy*, Vol. 2(6), 2004, pp. 34-41.

[61] S. Schrittwieser and S. Katzenbeisser, "Code obfuscation against static and dynamic reverse engineering", *Proceedings of 13th International Conference on Information Hiding*, 2011, pp. 45-60.

[62] D. Huang, "Anonymous Certification Services," *Proceedings of GLOBECOM'2010*, 2010, pp.1-6.

[63] Anonymizer, available at: http://www.anonymizer.com

[64] Lucent Personalized Web Assistant (LPWA), available at: http://www.bell-labs.com/projects/lpwa

[65] E. Martinez, G. Magoulas, S. Chen and R. Macredie, "Modeling human behavior in user-adaptive systems: Recent advances using soft computing techniques," *Expert Systems with Applications*, Vol. 29(2), 2005, pp. 320-329.

[66] A. Kobsa nd J. Schreck, "Privacy through pseudonymity in user-adaptive systems," *ACM Transactions on Internet Technology (TOIT)*, Vol. 3(2), 2003, pp. 149-183

[67] V. Shmatikov and M. Wang, "Measuring relationship anonymity in mix networks," *Proceedings of the 5th ACM Workshop on Privacy in Electronic Society*, 2006, pp. 59-62

[68] R. Dingledine, N. Mathewson, P. Syverson, "Tor: The second-generation onion router," *Proceedings of the 13th USENIX Security Symposium*, 2004, pp. 82-88.

[69] J. Reardon and I. Goldberg, "Improving Tor using a TCP-over-DTLS tunnel," *Proceedings of the 18th conference on USENIX security symposium*, 2009, pp. 39–41

[70] D. Chaum, A. Fiat and M. Naor, "Untraceable Electronic Cash," *Proceedings of Advances in Cryptology*, 1990, pp. 319-327.

[71] S. von Solms and D. Naccache, "On Blind Signatures and Perfect Crimes," *Computers Security*, Vol. 1, 1992, pp. 581-583.

[72] S. Malladi, J. Alves-Foss, and R. Heckendorn, "On preventing replay attacks on security protocols," *Proceedings of International Conference on Security and Management*, 2002, pp.77-83

[73] A. Pfitzmann and M. Kohntopp, "Anonymity, Unobservability, and Pseudonymity - A Proposal for Terminology," DIAU00, *Lecture Notes in Computer Science*, 2009, pp 1-9.

[74] S. S. Yau and Y. Yin, "A Privacy Preserving Repository for Data Integration across Data Sharing Services," *IEEE Transactions on Services Computing*, Vol. 1(3), 2008, pp. 130-140.

[75] W. Halang, "A function oriented design for industrial distributed real time control systems", *Proceedings of the 2nd International Conference on Software Engineering for Real Time Systems*, 1989, pp. 116-120.

[76] M. Adler, "An algebra for data flow diagram process decomposition", *IEEE Transactions on Software Engineering*, Vol. 14(2), 1988, pp. 169 - 183.

[77] S. S. Yau and H. G. An, "Software Engineering Meets Services and Cloud Computing," *IEEE Computer*, Vol. 44(10), 2011, pp. 47-53.

[78] G. Brahnmath, R. R. Raje, A. Olson, M. Auguston, B. R. Bryant, and C. C. Burt, "A Quality of Service Catalogue for Software Components", *Proceedings of Conference on Southeastern Software Engineering*, 2002, pp. 513–520.

[79] J. Ivan, H. Caroline and F. Stephane, "A Comprehensive Quality Model for Service-Oriented Systems", *Software Quality Control*, Vol. 17(1), 2009, pp. 65-98.

[80] C. Yang, C. Lin and S. Chen, "A Workflow-based Computational Resource Broker with Information Monitoring in Grids", *Proceedings of 5th International Conference on Grid and Cooperative Computing*, 2006, pp. 105-206.

[81] S. Venugopal, X. Chu and R. Buyya, "A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol", *Proceedings of 16th International Workshop on Quality of Service (IWQoS 2008),* 2008.

[82] C. T. Yang, P. C. Shih, C. F. Lin and S. Y. Chen, "A Resource Broker with an Efficient Network Information Model on Grid Environments", *The Journal of Supercomputing*, Vol. 40(3), 2007, pp. 249-267.

[83] D. Zhang and J. Xu, "Securing instance-level interactions in web services", *Proceedings of International Symposium on Autonomous Decentralized Systems*, 2005. pp. 443–450.

[84] C. Yang, C. Lin and S. Chen, "A Workflow-based Computational Resource Broker with Information Monitoring in Grids", *Proceedings of 5th International Conferences on Grid and Cooperative Computing*, 2006, pp. 105-206.

[85] S. Venugopal, X. Chu and R. Buyya, "A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol", *Proceedings of 16th International Workshop on Quality of Service (IWQoS 2008),* 2008, pp. 34-41.

[86] C. T. Yang, P. C. Shih, C. F. Lin and S. Y. Chen, "A Resource Broker with an Efficient Network Information Model on Grid Environments", The Journal of Supercomputing, vol. 40(3), 2007, pp. 249-267.

[87] R. Buyya, C. S. Yeo, and S. Venugopal, "Market Oriented Cloud Computing: Vision, Hype, and Reality for Delivering IT Services as Computing Utilities", *Proceedings of 10th IEEE International Conference on High Performance Computing and Communications*, 2008, pp. 234-242.

[88] P. Lardieri, J. Balashbramanian and D. C. Schmidt, "A multi-layered resource management framework for dynamic resource management in enterprise DRE systems", *Journal of Systems and Software*, Vol. 80(7), 2007, pp. 984-996.

[89] P. E. Hladik, H. Cambazard, A. M. Deplanche and N. Jussien, "Solving a real-time allocation problem with constraint programming", *Journal of Systems and Software*, Vol. 81(1), 2008, pp. 132-149.

[90] J. N. Hooker and G. Ottoson, "Logic-based benders decomposition", *Mathematical Programming*, Vol. 96, 2003, pp. 33-60.

[91] J. Stoesser, C. Roessle and D. Neumann, "Decentralized Online Resource Allocation for Dynamic Web Service Applications", *Proceedings of 4th International Conferences on Enterprise Computing, E-Commerce, and E-Service*, 2007, pp. 425-428.

[92] M. Mazzucco, I. Mitrani, J. Palmer, M. Fisher and P. McKee, "Web Service Hosting and Revenue Maximization", *Proceedings of 5th European Conferences on Web Services (ECOWS'07)*, 2007, pp. 92-98.

[93] M. Kallits, G. Michailidis and M. Devetsikiotis, "Pricing and Optimal Resource Allocation in Next Generation Network Services", *Proceedings of IEEE Sarnoff Symposium*, 2007, pp. 1-5.

[94] F. Meshkati, H. V. Poor and S. C. Schwartz, "Energy-Efficient Resource Allocation in Wireless Networks", *Signal Processing Magazine*, Vol. 24(3), 2007, pp. 58-68.

[95] C. Bae and D. Cho, "Fairness-Aware Adaptive Resource Allocation Scheme in Multihop OFDMA Systems", *IEEE Communications Letters*, Vol. 11(2), 2007, pp. 134-136.

[96] S. Stuijk, T. Basten, M. Geilen and H. Corporaal, "Multiprocessor Resource Allocation for Throughput Constrained Synchronous Dataflow Graphs", *Proceedings of Conferences on the 44th IEEE annual Design Automation*, 2007, pp. 777-782.

[97] Z. Yang, N. Ye and Y.-C. Lai, "QoS model of a router with feedback control", *Quality and Reliability Engineering*, Vol. 22(4), 2006, pp. 429-444.

[98] N. Ye, B. Harish, X. Li and T. Farley, "Batch scheduled admission control for service dependability of computer and network resources", *Information, Knowledge, Systems Management*, Vol. 5(4), 2006, pp. 211-226.

[99] Q. Zhang, L. Cherkasova and E. Smirni, "A Regression Based Analytic Model for Dynamic Resource Provisioning of Multi-Tier Applications", *Proceedings of the 4th International Conference on Autonomic Computing*, 2007, pp. 27-36.

[100] S. S. Yau, Y. Yin and H. G. An, "An Adaptive Model for Tradeoff between Service Performance and Security in Service-based Environments", *Proceedings of International Conference on Web Services (ICWS 2009)*, 2009, pp. 287-294

[101] L. Eggert and J. Heidemann, "Application-Level Differentiated Services for Web Services", *Journal of World-Wide Web*, Vol. 2(3), 1999, pp. 133-142.

[102]. G. Rao and B. Ramamurthy, "DiffServer: Appliction Level Differentiated Services for Webservers," *Proceedings of IEEE International Conference on Communication*, Vol. 5, 2001, pp. 1633-1637.

[103] C. Lu, Y. Lu, T.F. Abdelzaher, J.A. Stankovic, and S.H. Son, "Feedback Control Architecture and Design Methodology for Service Delay Guarantees in Web Servers," *IEEE Transactions on Parallel and Distributed Systems,* Vol. 17(9), 2006, pp 1014-1027.

[104] T.F. Abdelzaher, J.A. Stankovic, C. Lu, R. Zhang, and Y. Lu, "Feedback Performance Control in Software Services," *IEEE Control Systems Magazine*, Vol. 23(3), 2003, pp. 74–90.

[105] G. Wang, A. Chen, C. Wang, C. Fung and S. Uczekaj, "Integrated quality of service (QoS) management in service-oriented enterprise architectures", *Proceedings of the 8th International Conference on Enterprise Distributed Object Computing*, 2004, pp. 21-32.

[106] S. S. Yau, N. Ye, H. Sarjoughian and D. Huang, "Developing Service-based Software Systems with QoS Monitoring and Adaptation", *Proceedings of the 12th International Workshop on Future Trends of Distributed Computing Systems*, 2008, pp. 74-80.

[107] S. S. Yau, N. Ye, H. Sarjoughian, D. Huang, A. Roontiva, M. Baydogan and M. Muqsith, "Towards Development of Adaptive Service-based Software Systems", *IEEE Transactions on Service Computing*, 2009.

[108] I. Griva, S. G. nash and A. Sofer, "Linear and Nonlinear Optimazation (2nd ed.)", Society for Industrial Mathematics, Philadelphia, PA, 2008

[109] S. Kona, A. Bansal, G. Gupta, and T. Hite, "Web Service Discovery and Composition using USDL", *Proceedings of 3rd IEEE International Conference on E-Commerce Technology*, 2006, pp. 65-67.

[110] M.L. Massie, B.N. Chun and D.E. Culler, "Ganglia Distributed Monitoring System: Design, Implementation, and Experience", *Parallel Computing,* Vol. 30, 2004, pp. 817–840.

[111] C. Yang, C. Lin and S. Chen, "A Workflow-based Computational Resource Broker with Information Monitoring in Grids", *Proceedings of 5th International Conference on Grid and Cooperative Computing*, 2006, pp. 105-206

[112] S. Venugopal, X. Chu and R. Buyya, "A Negotiation Mechanism for Advance Resource Reservation using the Alternate Offers Protocol", *Proceedings of 16th International Workshop on Quality of Service (IWQoS 2008),* 2008.

[113] I. Griva, S. G. nash and A. Sofer, "Linear and Nonlinear Optimazation (2nd ed.)", Society for Industrial Mathematics, Philadelphia, PA, 2008

[114] S. S. Yau, N. Ye, H. Sarjoughian, D. Huang, A. Roontiva, M. Baydogan and M. Muqsith, “Towards Development of Adaptive Service-based Software Systems”, *IEEE Transactions on Service Computing*, 2009

[115] C. T. Yang, P. C. Shih, C. F. Lin and S. Y. Chen, “A Resource Broker with an Efficient Network Information Model on Grid Environments”, *The Journal of Supercomputing*, Vol. 40(3), 2007, pp. 249-267

[116] S. S. Yau, N. Ye, H. Sarjoughian and D. Huang, “Developing Service-based Software Systems with QoS Monitoring and Adaptation”, *Proceedings of 12th International Workshop on Future Trends of Distributed Computing Systems*, 2008, pp. 74-80

[117] S. S. Yau, N. Ye, H. Sarjoughian, D. Huang, A. Roontiva, M. Baydogan and M. Muqsith, “Towards Development of Adaptive Service-based Software Systems”, *IEEE Transactions on Service Computing*, 2009

[118] I. Griva, S. G. nash and A. Sofer, “Linear and Nonlinear Optimazation (2nd ed.)”, Society for Industrial Mathematics, Philadelphia, PA, 2008

[119] D. Spielman and S. Teng, "Smoothed analysis of algorithms: why the simplex algorithm usually takes polynomial time", *Proceedings of 33th Annual ACM Symposium on Theory of Computation*, 2001, pp. 296-305